





DUDLEY KNOX LIBRARY  
NAVAL POSTGRADUATE SCHOOL  
MONTEREY, CALIF 93940













# NAVAL POSTGRADUATE SCHOOL

## Monterey, California



# THESIS

ADVANCED SIMULATION OF DIGITAL FILTERS

by

Gerald S. Doyle

September 1980

Thesis Advisor:

D. E. Kirk

Approved for public release; distribution unlimited

T196585





| REPORT DOCUMENTATION PAGE   |                       | READ INSTRUCTIONS<br>BEFORE COMPLETING FORM                              |
|---|-----------------------|--|
| 1. REPORT NUMBER  | 2. GOVT ACCESSION NO. | 3. RECIPIENT'S CATALOG NUMBER  |
| 4. TITLE (and Subtitle)<br>Advanced Simulation of Digital Filters   |                       | 5. TYPE OF REPORT & PERIOD COVERED<br>Master's Thesis;<br>September 1980 |
|   |                       | 6. PERFORMING ORG. REPORT NUMBER   |
| 7. AUTHOR(s)<br>Gerald S. Doyle   |                       | 8. CONTRACT OR GRANT NUMBER(s)   |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS<br>Naval Postgraduate School<br>Monterey, California 93940  |                       | 10. PROGRAM ELEMENT, PROJECT, TASK<br>AREA & WORK UNIT NUMBERS           |
| 11. CONTROLLING OFFICE NAME AND ADDRESS<br>Naval Postgraduate School<br>Monterey, California 93940  |                       | 12. REPORT DATE<br>September 1980  |
|   |                       | 13. NUMBER OF PAGES<br>307   |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)   |                       | 15. SECURITY CLASS. (of this report)<br>UNCLASSIFIED                     |
|   |                       | 15a. DECLASSIFICATION/DOWNGRADING<br>SCHEDULE                            |
| 16. DISTRIBUTION STATEMENT (of this Report)<br>Approved for public release; distribution unlimited  |                       |  |
| 17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)  |                       |  |
| 18. SUPPLEMENTARY NOTES   |                       |  |
| 19. KEY WORDS (Continue on reverse side if necessary and identify by block number)<br>Digital Filter; Interactive Graphics; Z-Plane   |                       |  |
| 20. ABSTRACT (Continue on reverse side if necessary and identify by block number)<br>An Advanced Simulation of Digital Filters has been implemented on the IBM 360/67 computer utilizing Tektronix hardware and software. The program package is appropriate for use by persons beginning their study of digital signal processing or for filter analysis. The ASDF programs provide the user with an interactive method by which filter pole and zero locations can be manipulated. Graphical output on both the Tektronix graphics screen and the Versatec plotter are provided to observe the effects of pole-zero movement. |                       |  |





Approved for public release; distribution unlimited.

Advanced Simulation of Digital Filters

by

Gerald S. Doyle  
Captain, United States Army  
B.S., United States Military Academy, 1973

Submitted in partial fulfillment of the  
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL  
September, 1980

---





## ABSTRACT

An Advanced Simulation of Digital Filters has been implemented on the IBM 360/67 computer utilizing Tektronix hardware and software. The program package is appropriate for use by persons beginning their study of digital signal processing or for filter analysis. The ASDF programs provide the user with an interactive method by which filter pole and zero locations can be manipulated. Graphical output on both the Tektronix graphics screen and the Versatec plotter are provided to observe the effects of pole-zero movement.



## TABLE OF CONTENTS

|      |   |    |
|------|---|----|
| I.   | INTRODUCTION . . . . .                        | 7  |
| II.  | SELECTION OF ASDF ALGORITHMS . . . . .        | 8  |
|      | A. ALGORITHMS SELECTED . . . . .              | 9  |
|      | B. DEFINING THE DATA STRUCTURE . . . . .      | 9  |
| III. | IMPLEMENTED SOLUTION . . . . .                | 11 |
|      | A. INTERFACE WITH THE Z-PLANE . . . . .       | 11 |
|      | B. TIME RESPONSE . . . . .                    | 19 |
|      | C. FREQUENCY RESPONSE . . . . .               | 27 |
|      | D. OUTPUT PLOTS . . . . .                     | 29 |
|      | E. SUPERVISORY ROUTINES . . . . .             | 30 |
| IV.  | STRUCTURE OF ASDF COMMANDS . . . . .          | 34 |
|      | A. SUMMARY OF DATA INPUT MODES . . . . .      | 37 |
|      | 1. Interactive Graphics (IAG) . . . . .       | 37 |
|      | 2. Rectangular . . . . .                      | 38 |
|      | 3. Polar . . . . .                            | 38 |
|      | B. THE POLZRO COMMAND SUBSTRUCTURE . . . . .  | 39 |
|      | C. SUMMARY OF POLZRO SUBCOMMANDS . . . . .    | 43 |
| V.   | LOADING THE ASDF PACKAGE . . . . .            | 44 |
|      | A. LOGIN PROCEDURE . . . . .                  | 44 |
|      | B. TESTING OSX001 FOR ASDF PROGRAMS . . . . . | 45 |
|      | 1. ASDF is on OSX001 . . . . .                | 46 |
|      | 2. ASDF is not on OSX001 . . . . .            | 46 |





|             |   |     |
|-------------|---|-----|
| VI.         | USING ASDF . . . . .                            | 50  |
| A.          | EXAMPLE ONE - INTERACTIVE GRAPHICS . . . . .    | 50  |
| B.          | EXAMPLE TWO - RECTANGULAR MODE . . . . .        | 59  |
| C.          | EXAMPLE THREE - POLAR MODE . . . . .            | 80  |
| VII.        | ADDITIONAL EXAMPLES . . . . .                   | 94  |
| VIII.       | INTERPRETATION OF ASDF OUTPUT . . . . .         | 201 |
| IX.         | SUMMARY AND CONCLUSIONS . . . . .               | 212 |
| APPENDIX A: | PROBABLE SOLUTIONS FOR DIFFICULTIES. . . . .    | 216 |
| APPENDIX B: | IMPORTANT SUBROUTINES AND VARIABLES. . . . .    | 218 |
| APPENDIX C: | STRUCTURE FOR ASDF TEXT FILE . . . . .          | 224 |
| APPENDIX D: | JCL FOR EXEC HRD\$CPY COMMAND . . . . .         | 226 |
| APPENDIX E: | SOURCE DECK FOR ASDF191 EXEC . . . . .          | 227 |
| APPENDIX F: | SOURCE DECK FOR ASDF192 EXEC . . . . .          | 229 |
| APPENDIX G: | SOURCE DECK FOR ASDF COMMAND: POLZRO. . . . .   | 233 |
| APPENDIX H: | SOURCE DECK FOR ASDF COMMAND: RESPONSE. . . . . | 276 |
| APPENDIX I: | SOURCE DECK FOR ASDF COMMAND: HRD\$CY. . . . .  | 289 |
|             | LIST OF REFERENCES. . . . .                     | 306 |
|             | INITIAL DISTRIBUTION LIST . . . . .             | 307 |



## ACKNOWLEDGEMENTS

The author wishes to acknowledge the people who made this effort possible. Of greatest importance is the love and guidance provided by his parents, Margaret and C. E. Doyle, and sister, Evelyn. It is to them that this thesis is dedicated. Special mention goes to Professor Richard Hamming for his insight and explanations of finite precision effects in digital filters. Thanks are due to Professors D. E. Kirk and R. D. Strum for their suggestions and comments in structuring this work. For their assistance with system problems, thanks are due to J. Foust, F. Wheeler, R. Donat, and H. Doleman. Moreover, thanks are due to E. Christian, and M. F. Bradley for their assistance in preparing the manuscript. Of no less importance are the evening computer center operators J. Kallweit, M. Anderson, and E. Donnellan for their consideration and assistance in processing the figures. However, the author alone is responsible for any shortcomings in this thesis, for the ultimate choice of what was done was his.





## 1. INTRODUCTION

Digital methods have recently begun to dominate the selection of transmission and filtering schemes. When the student first confronts the discrete domain, he encounters a series of unfamiliar concepts. While most of the discrete concepts have parallels in the continuous or s-domain, this dualism is not apparent initially.

The programs comprising the Advanced Simulation of Digital Filters (ASDF) provide the user with a convenient method for developing intuition in the discrete domain. This is achieved by allowing the user to generate digital filters by directly entering the poles and zeros into a graphics terminal. The ASDF programs then perform the calculations required to display the characteristics of the user generated filter.

The overriding objective of the ASDF program package is to provide the user with an interactive method for modifying the filter pole/zero locations in the z-plane, as well as to compute the filter responses. Both time and frequency responses are computed and displayed graphically on the Tektronix 4012 graphics terminal. [Ref. 9]



## II. SELECTION OF ASDF ALGORITHMS

There are several parameters which are particularly important in selecting the algorithms used in the ASDF programs. Since one of the primary objectives of these programs is to provide the user with clear explanations of methods for computing digital filter characteristics, direct or straightforward methods should be used. The importance of developing a user's intuition for the discrete domain and digital filters suggests that graphical rather than numerical outputs would be most appropriate. The use of graphical outputs to characterize a filter's responses implies minimum importance of precision in these results. Since the reason for computing the filter characteristics is to develop the user's intuition for the discrete domain, there are no pressing constraints on the selected algorithms with regard to computational efficiency, speed, or memory space utilization. In fact, the final ASDF package spends the greatest portion of its execution time making the interface with the  $z$ -plane, and the form of the output easy to understand and to use. The greatest part of the memory space utilized for the ASDF programs is consumed in the interfacing processes rather than in the methods by which the data is stored, used, or computed.





## A. ALGORITHMS SELECTED

Because of the above stated objectives, a simple algorithm was selected for computation of time domain responses of digital filters. Once the user enters the desired pole and zero locations, a direct-form-two difference equation representation of the filter is generated [Ref. 5]. The unit sample sequence is used for the filter input, the difference equations are solved recursively in the discrete time domain and the unit sample response is recorded as the output. The unit step response is calculated by a similar method.

It is known that the direct-form-two filter implementation is very sensitive to parameter variations, and as such cannot be utilized in situations requiring extreme accuracy. The advantages associated with simplicity of understanding the algorithms, the ease of programming the general case, and the deemphasis of accuracy in outputs for marginally stable filters outweighs this disadvantage.

## B. DEFINING THE DATA STRUCTURE

A detailed description of the important variables in the ASDF programs is given in Appendix B. There is, however, an overriding data structure common to all portions of the ASDF package which is important.

The array POLZRO holds the locations of the poles and zeros of the filter under analysis. In all cases, zeros are





stored in POLZRO(I,J) for values of I from one to ten. Pole locations are stored in POLZRO(I,J) for values of I from eleven to twenty. Only the top half plane pole of each complex pair is stored. There are five sections of POLZRO(I,J), i.e. values of J from one to five. POLZRO(I,1) is the real part of the rectangular representation of the ith root. POLZRO(I,2) is the imaginary part of the rectangular representation of the ith root. POLZRO(I,3) and POLZRO(I,4) are the magnitude and angle portions of the ith root location in polar form. POLZRO(I,5) is the root type designator. Root type designators are: one for real zeros, two for complex zeros, three for real poles, and four for complex poles.

The array IROOTS(I) holds the numbers of roots of each type of root already entered into the system. IROOTS(1) and IROOTS(3) hold the number of real zeros and poles, respectively. IROOTS(2) and IROOTS(4) hold the number of complex zero and pole pairs, respectively.

The variable IERROR is the error flag, which is set whenever the user generates an error. There are numerous possible errors. These include trying to delete roots which have not been entered, or adding roots which cause the maximum system order to be exceeded.



### III. IMPLEMENTED SOLUTION

Having determined the general form of the solution it remained to generate the software implementation. There were five major areas to be programmed: the interface with the z-plane, the time response, the frequency response, the plotting of output curves, and the supervisory routines.

#### A. INTERFACE WITH THE Z-PLANE

Since the method by which the user modifies the pole and zero locations in the z-plane is of no theoretical interest, the discussion of this implementation will be brief. Extensive coverage of the subroutines employed and the definitions of the variables used is reserved for Appendix B.

The ASDF command which allows the user to make pole-zero modifications in the z-plane is POLZRO. This command initiates execution of a load module which is structured as depicted in figure 3-1. Each of the boxes shown represents a subroutine. Lines with arrow heads on both ends indicate that program control passes to the subroutine and then directly back to the calling routine with no intermediate branches. The main program, MAIN, initializes the required arrays, and tests to ensure that the user has generated a realizable causal filter.





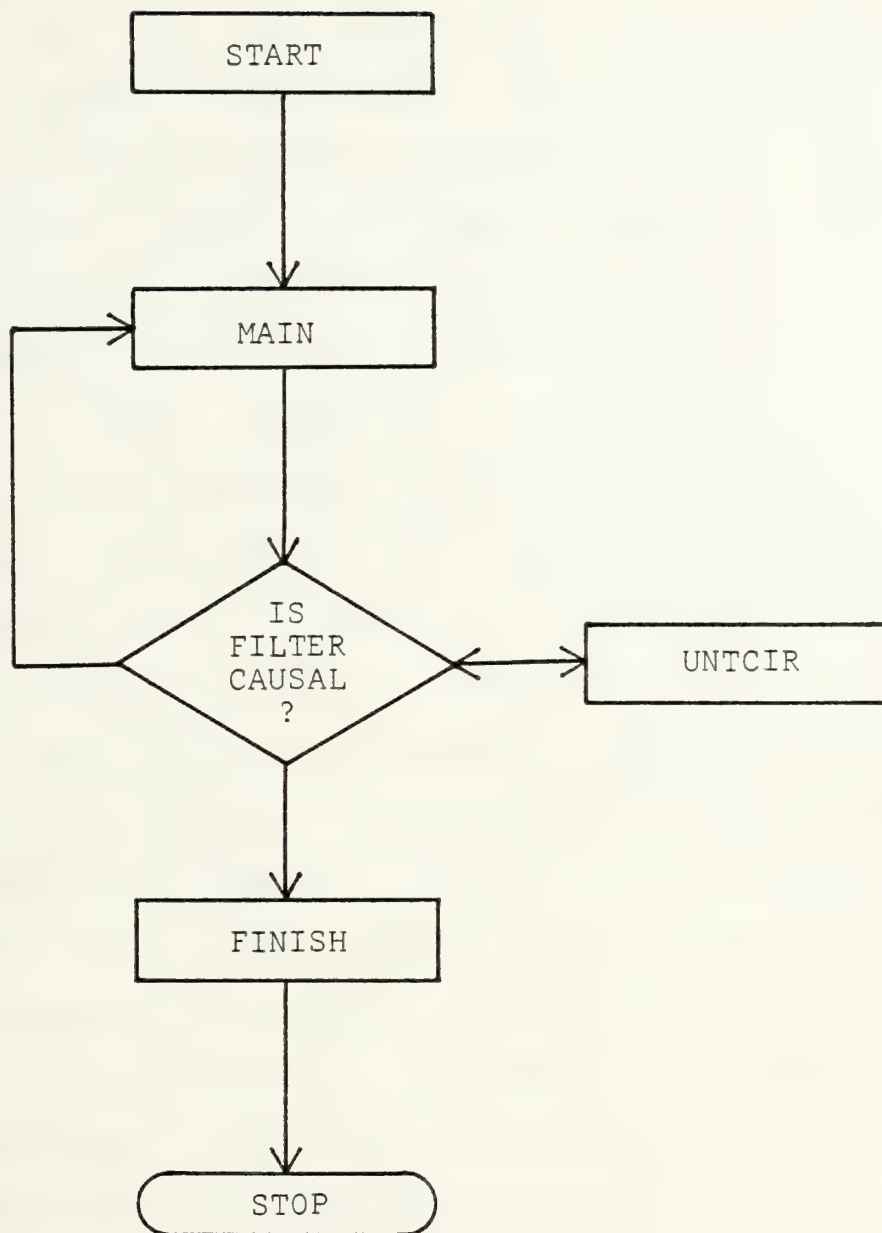


Figure 3-1 Flowchart Of POLZRO



The actual interaction with the z-plane is accomplished in subroutine UNTCIR. The structure of UNTCIR is diagrammed in figure 3-2. The large double ended arrow indicates that there are many subroutines of the form CRRMMM. The subroutines LETTER and BOXUC generate the unit circle and command array. PLOT-10 software [Ref. 10] statements determine the location of the user positioned cursor. This cursor position is adjusted with software tabs to one of twenty-six possible locations. Based on the X and Y location of the adjusted cursor position a specific subroutine is selected for execution. The flashing alphanumeric cursor is positioned in the selected command box using the adjusted cursor location as a reference. The subroutine then reads two alphanumeric characters typed by the user. The combination "space", "e" indicates that the user wishes the selected subroutine to be executed. The combination "space", "x" indicates that the user wishes to terminate the pole zero manipulation phase of the program.

All of the commands which change the status of the z-plane have names of the form CRRMMM where C is either "A" for add root(s) or "D" for delete root(s). The RR indicates root type: RZ - real zero, CZ - complex zero, RP - real pole, CP - complex pole. The MMM stands for the mode of data entry. IAG stands for interactive graphics i.e. the roots will be adjusted with the cursor. POL and RCT stand for polar and rectangular, respectively. The general





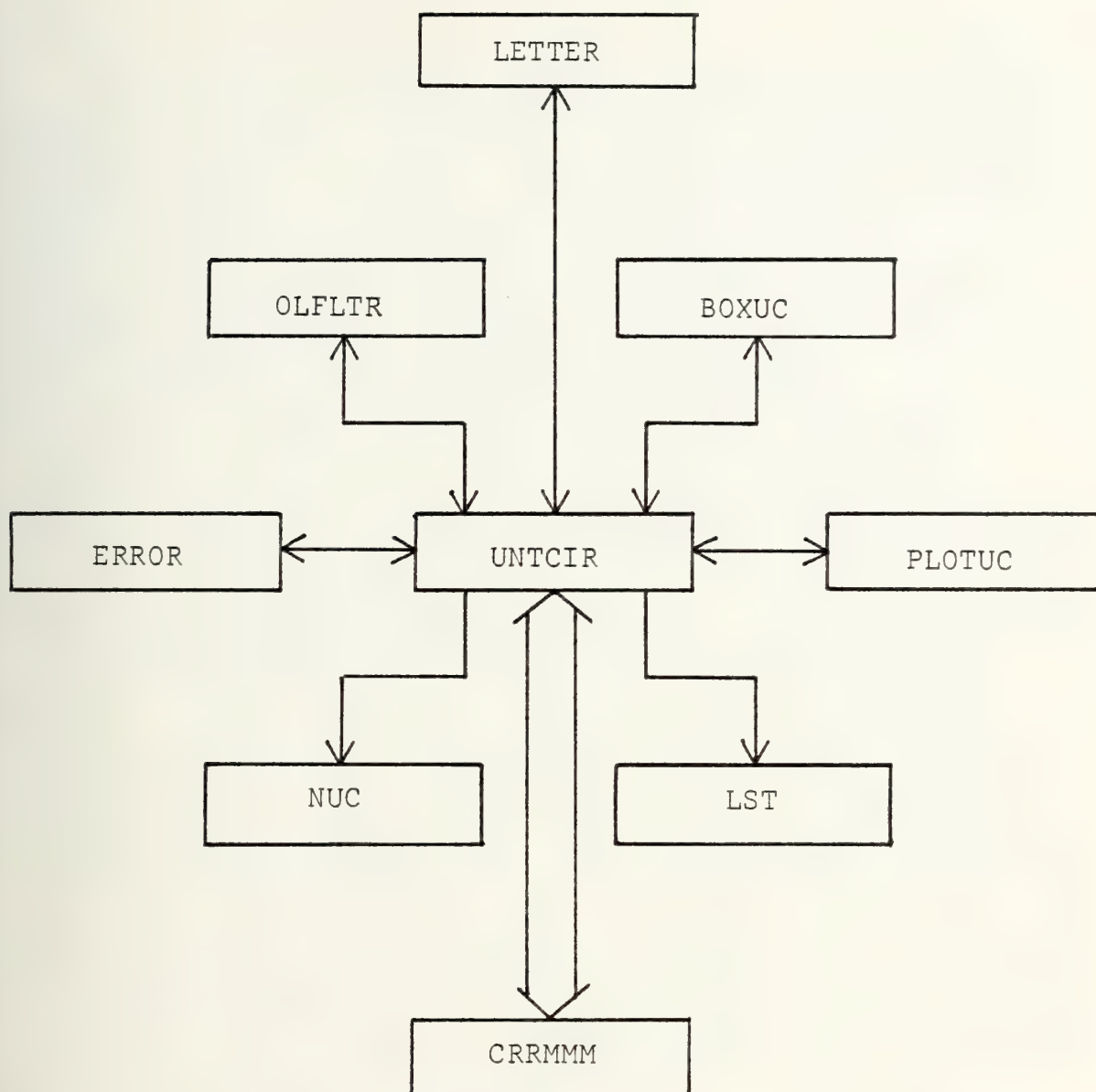


Figure 3-2 Structure Of Subroutine UNTCIR



structure of the CRRMMM subroutines is described in figure 3-3.

All CRRMMM subroutines follow the same general pattern. The POLZRO table is checked to determine whether or not the issued command is valid. Poles and zeros must be present before they can be deleted. There cannot be more than ten poles or ten zeros. The desired pole or zero location is next acquired. For subroutines adding roots<sup>1</sup> the entered location is that of the root(s) to be added. For subroutines deleting roots the entered location is any location near the root(s) to be deleted. Using subroutine LOCATE, the POLZRO table is searched for the root of the correct type which is nearest the specified location. This nearest root is then highlighted. In all cases, except when adding roots in the IAG mode, the user is asked whether or not the correct location has been determined. Once the root location is verified the root is added and plotted with subroutines STRRTS and PLOTRT. For subroutines deleting roots the POLZRO table is adjusted with subroutine UPDATE and the unit circle is redrawn.

The user may make an error in carrying out the above steps, i.e., by placing a pole outside the unit circle, by exceeding the maximum system order, or by entering a zero outside ten units from the origin. In each of these cases an

---

<sup>1</sup> Hereafter, poles and zeros will be referred to as roots.



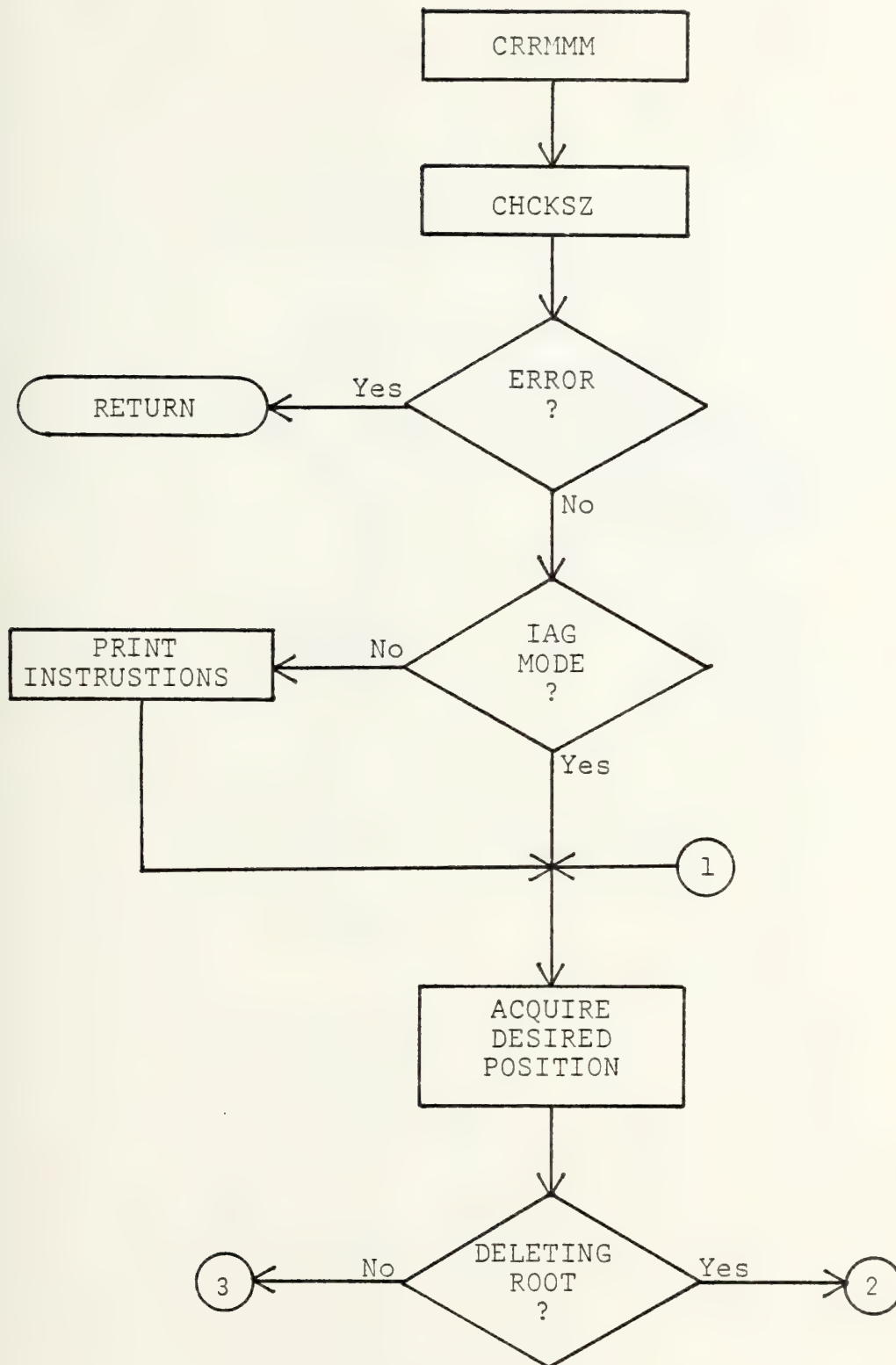


Figure 3-3a Structure of CRRMMM Subroutines





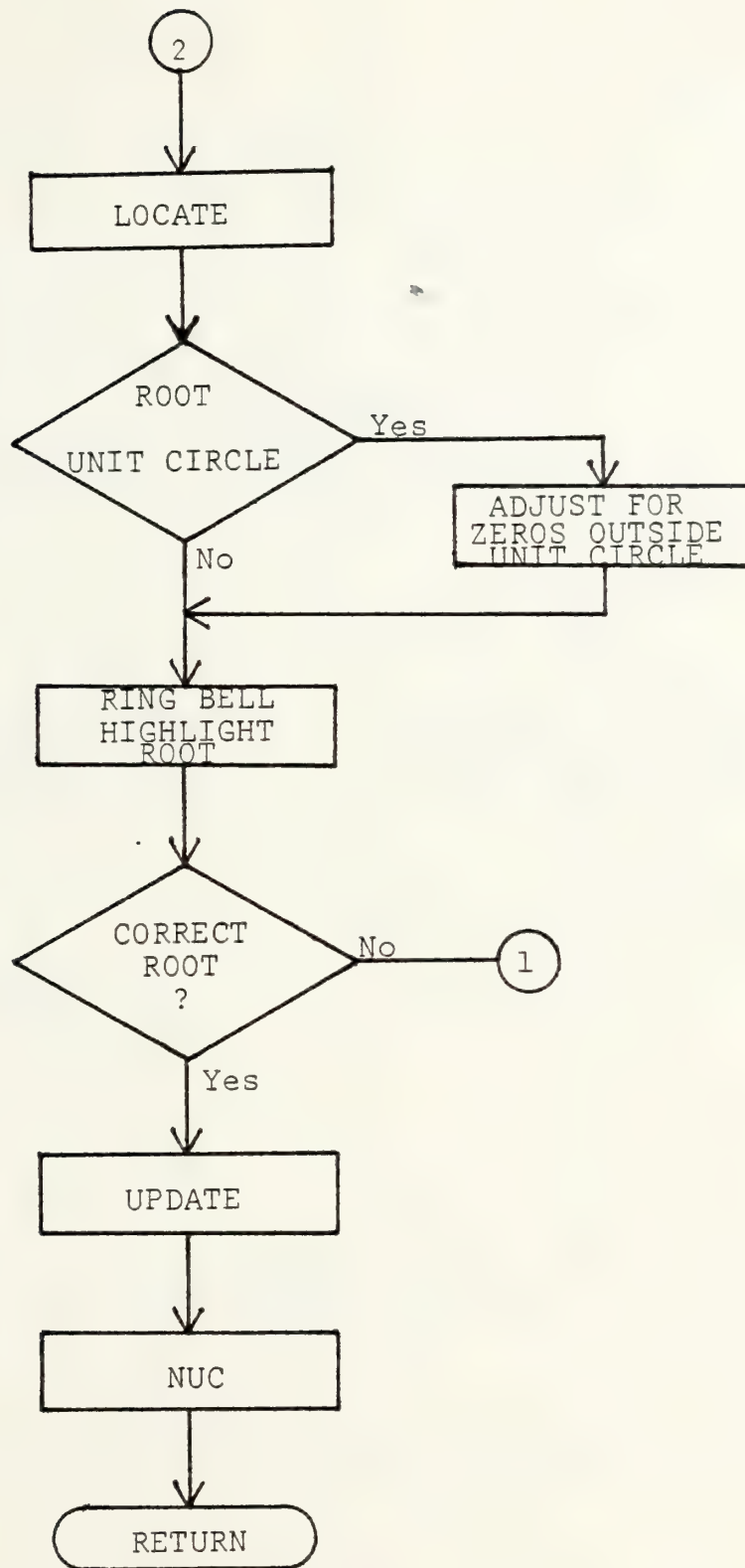


Figure 3-3b Structure Of CRRMMM Subroutines Continued



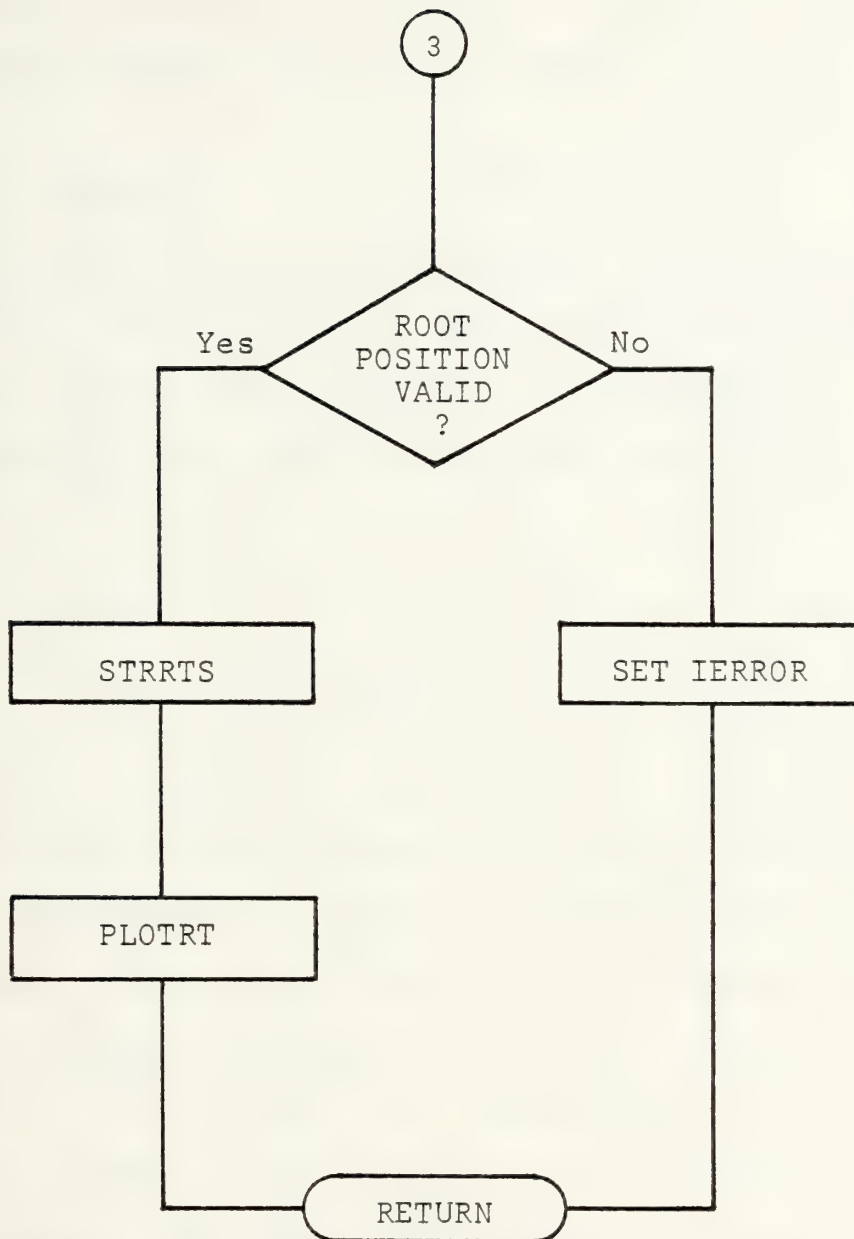


Figure 3-3c Structure Of CRRMMM Subroutines Continued





error flag is set. Subroutine ERROR handles the printing of messages and subroutine LST prints a listing of the current filter status before the program resumes.

## B. TIME RESPONSE

The transfer function of a digital filter is defined in equation 3.1. The  $z_i$ ,  $p_i$ ,  $a_k$ , and  $b_r$  will be defined in a subsequent section. The variable A is an arbitrary constant assigned by the user and is called the filter gain.

$$H(z) = \frac{A \prod_{i=1}^M (z - z_i)}{\prod_{i=1}^N (z - p_i)} = \frac{\sum_{r=0}^M b_r z^{r-N}}{1 - \sum_{k=1}^N a_k z^{-k}} \quad (3.1)$$

Inherent in generating the time domain response with a direct-form-two filter implementation are the determination of the coefficients  $a_k$  and  $b_r$  in equation (3-1), and the computation of the unit sample and unit step sequences. Having determined the filter coefficients, the time domain response of the filter is determined by using the appropriate filter input sequence and recording the output of the filter for the desired number of points. This is accomplished by recursively solving a difference equation which characterizes the filter for 1024 time points. The standard direct-form-two filter implementation is shown in figure 3-4. The  $z^{-1}$  terms represent unit time delays. This



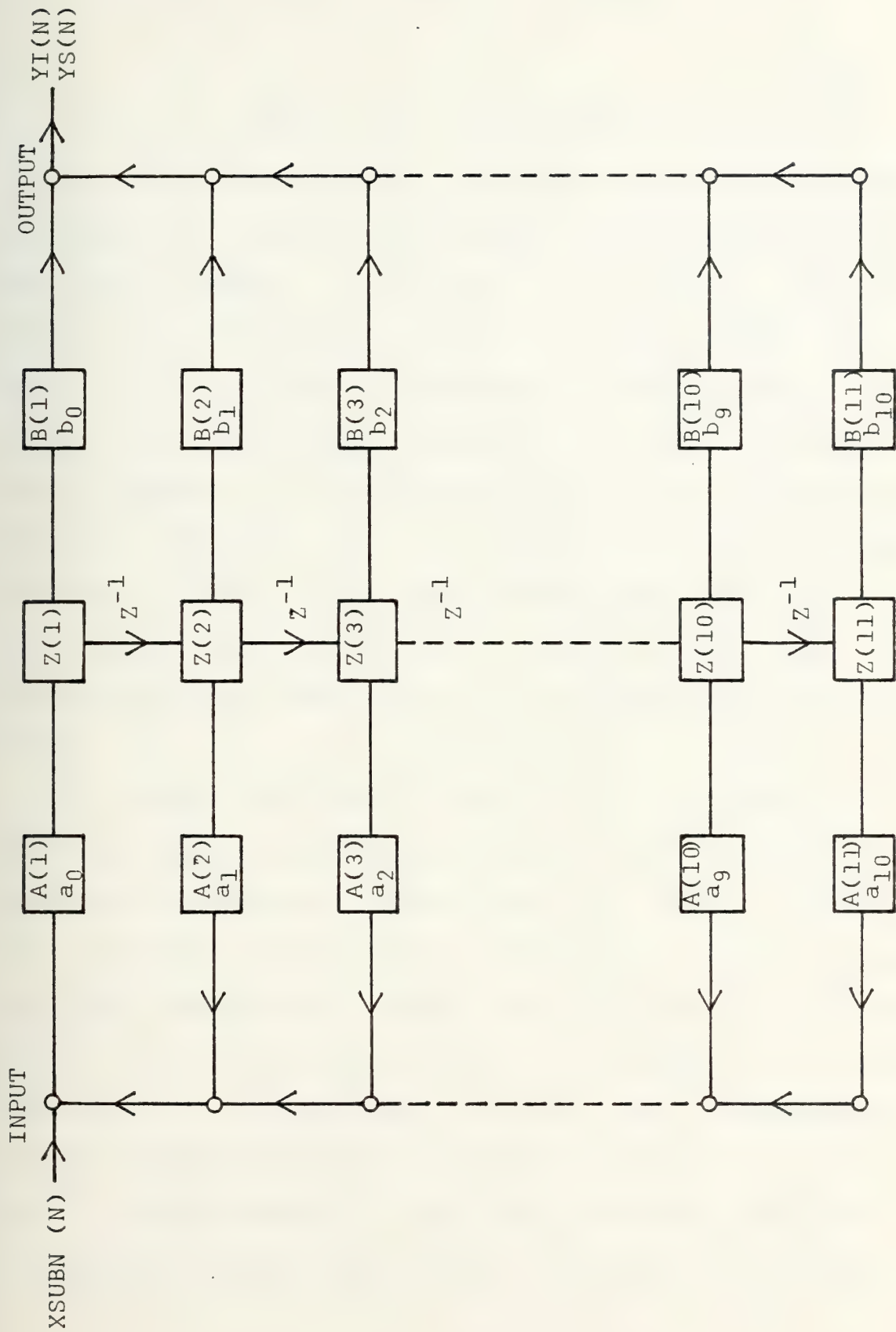


Figure 3-4 Direct Form Two Filter Implementation



figure corresponds to the difference equations:

$$y(n) = \sum_{k=1}^N a_k y(n-k) + \sum_{r=0}^N b_r x(n-N+r) \quad (3.2)$$

To compute the filter's time response, three operations are required: calculate the  $a_k$  and  $b_r$  coefficients, generate the input sequence  $x(k)$ ,  $k=0, 1, \dots, 1024$ , solves the difference equations and stores the filter output values. There are four central subroutines which perform these operations: ASUBK, BSUBR, RSPNSE, and COEFF. Subroutines ASUBK, BSUBR, and COEFF generate the filter coefficients starting with the pole and zero locations specified. Subroutine RSPNSE solves the difference equations by coordinating the other three routines, generating the appropriate input sequence, and storing the output sequence values.

The ASUBK and BSUBR subroutines are identical in structure and purpose. ASUBK computes the degree of the denominator polynomial as two times the number of complex pole pairs plus the number of real poles. A complex vector is formed named POLES, which holds a listing of all the poles of the filter under analysis. The POLZRO array stores real poles in the form  $X+j0$ . Only the top half plane pole of each complex pole pair is stored, i.e.,  $(X+jY)$ , where  $X$  and  $Y$  are non-negative. The vector POLES has entries for all real poles and both poles of each complex pole pair.





Once the POLES vector has been generated the subroutine COEFF multiplies out the factors of the form  $(z-p_i)$  to generate the coefficients of the denominator polynomial.

The algorithm for multiplying out the denominator polynomial is essentially the same as would be done by hand. multiplication of a series of literal factors of the form  $(z-p_i)$ . The vector POLES is passed to the COEFF subroutine as the vector V. Since the denominator factors are of the form  $(z-p_i)$ , the V vector is changed to hold the values  $-p_i$ . If the system is of order less than ten, the remaining positions of V are filled with zeros. The degree of the denominator polynomial is subsequently used to determine how many poles are actually located at the origin.

The vectors used in the subroutine COEFF hold the coefficients for the polynomial being generated. The literal powers of "z" are suppressed. The vector H1 always contains the partial product of the K factors previously multiplied together. The vector H2 represents the old partial product times the "z" of the  $(z-p_i)$  factor. Since the coefficient of "z" is always one, H2 is a left shifted version of H1. The vector H3 represents the partial product times the next pole location  $(p_i)$ . This next  $-p_i$  value is stored in V(1). The new H1 vector is formed by adding the H2 and H3 vectors. H1 now contains the product of the first K+1 factors. The values in the V vector are next shifted such that the next pole location to be multiplied is always



in the V(1) position. These four steps are repeated ten times to generate the coefficients of the complete polynomial in "z". The formula for the transfer function requires coefficients of the denominator polynomial in terms of  $z^{-1}$  so the order of the coefficients is reversed.

Figure 3-5 depicts several iterations of the algorithm used in the COEFF subroutine. Figure 3-5a shows the initial conditions for each of the vectors defined in the above paragraph. Figure 3-5c shows the values of the vectors after the first iteration. Figure 3-5f shows the vector values after the second iteration. The third and subsequent iterations follow this pattern.

In precisely the same way as described above, the  $a_k$  coefficients are generated with the ASUBK subroutine. The zero locations are stored in POLZRO exactly as are the poles. The generation of the vector ZEROS and the use of the subroutine COEFF parallels the BSUBR subroutine exactly.

The significant variables in subroutine RSPNSE are: XSUBN, INPUT, OUTPUT, FLTRGN, A, and B. The vectors A and B hold the already computed values of  $a_k$  and  $b_k$ . A(1) holds  $a_0$ ; A(2) holds  $a_1$  . . . A(11) holds  $a_{10}$ . The  $b_r$  are stored similarly in the B vector. XSUBN is the current value of the input sequence. INPUT is the value of the node so labeled in figure 3-4. OUTPUT is similarly noted on figure 3-4. FLTRGN is the value of the filter gain, and is assigned by the user. The vector Z holds the delayed INPUT values.



|    |   |   |   |   |   |   |   | V      |
|----|---|---|---|---|---|---|---|--------|
| H1 | . | . | . | 0 | 0 | 0 | 1 | $-p_1$ |
| H2 | . | . | . | 0 | 0 | 0 | 0 | $-p_2$ |
| H3 | . | . | . | 0 | 0 | 0 | 0 | $-p_3$ |
|    |   |   |   |   |   |   |   | $-p_4$ |
|    |   |   |   |   |   |   |   | .      |
|    |   |   |   |   |   |   |   | .      |
|    |   |   |   |   |   |   |   | .      |

Figure 3-5a. Initial Conditions Of COEFF Vectors.

|    |   |   |   |   |   |   |        | V      |
|----|---|---|---|---|---|---|--------|--------|
| H1 | . | . | . | 0 | 0 | 0 | 1      | $-p_1$ |
| H2 | . | . | . | 0 | 0 | 1 | 0      | $-p_2$ |
| H3 | . | . | . | 0 | 0 | 0 | $-p_1$ | $-p_3$ |
|    |   |   |   |   |   |   |        | $-p_4$ |
|    |   |   |   |   |   |   |        | .      |
|    |   |   |   |   |   |   |        | .      |
|    |   |   |   |   |   |   |        | .      |

Figure 3-5b. Values Of COEFF Vectors Before The End Of The First Iteration.





|    |   |   |   |   |   |   |        |        |
|----|---|---|---|---|---|---|--------|--------|
|    |   |   |   |   |   |   |        | V      |
| H1 | . | . | . | 0 | 0 | 1 | $-p_1$ | $-p_2$ |
| H2 | . | . | . | 0 | 0 | 1 | 0      | $-p_3$ |
| H3 | . | . | . | 0 | 0 | 0 | $-p_1$ | $-p_4$ |
|    |   |   |   |   |   |   |        | $-p_5$ |
|    |   |   |   |   |   |   |        | .      |
|    |   |   |   |   |   |   |        | .      |
|    |   |   |   |   |   |   |        | .      |

Figure 3-5c. Values Of COEFF Vectors starting The Second Iteration.

|    |   |   |   |   |   |        |            |        |
|----|---|---|---|---|---|--------|------------|--------|
|    |   |   |   |   |   |        |            | V      |
| H1 | . | . | . | 0 | 0 | 1      | $-p_1$     | $-p_2$ |
| H2 | . | . | . | 0 | 1 | $-p_1$ | 0          | $-p_3$ |
| H3 | . | . | . | 0 | 0 | $-p_2$ | $+p_1 p_2$ | $-p_4$ |
|    |   |   |   |   |   |        |            | $-p_5$ |
|    |   |   |   |   |   |        |            | .      |
|    |   |   |   |   |   |        |            | .      |
|    |   |   |   |   |   |        |            | .      |

Figure 3-5d. Values Of COEFF Vectors before The End Of The Second Iteration.



|    |   |   |   |   |                |             | V      |
|----|---|---|---|---|----------------|-------------|--------|
| H1 | . | . | . | 1 | $-(p_1 + p_2)$ | $(p_1 p_2)$ | $-p_3$ |
| H2 | . | . | . | 1 | $-p_1$         | 0           | $-p_4$ |
| H3 | . | . | . | 0 | $-p_2$         | $(p_1 p_2)$ | $-p_5$ |
|    |   |   |   |   |                |             | $-p_6$ |
|    |   |   |   |   |                |             | .      |
|    |   |   |   |   |                |             | .      |
|    |   |   |   |   |                |             | .      |

Figure 3-5e. Values Of COEFF Vectors Starting The Third Iteration.

|    |   |   |                |                   |                  |  | V      |
|----|---|---|----------------|-------------------|------------------|--|--------|
| H1 | . | . | 1              | $-(p_1 + p_2)$    | $(p_1 p_2)$      |  | $-p_3$ |
| H2 | . | 1 | $-(p_1 + p_2)$ | $(p_1 p_2)$       | 0                |  | $-p_4$ |
| H3 | . | 0 | $-p_3$         | $+(p_1 + p_2)p_3$ | $-(p_1 p_2 p_3)$ |  | $-p_5$ |
|    |   |   |                |                   |                  |  | $-p_6$ |
|    |   |   |                |                   |                  |  | .      |
|    |   |   |                |                   |                  |  | .      |
|    |   |   |                |                   |                  |  | .      |

Figure 3-5f. Values Of COEFF Vectors before The End Of The Third Iteration.



Z(1) is no delay; Z(2) is delayed one time unit; Z(3) is delayed two time units, etc.

The subroutine RSPNSE generates the appropriate input sequence. For the unit sample response the input sequence is 1, 0, 0, 0, . . . . For the unit step response the input sequence is 1, 1, 1, 1, . . . . The appropriate value of XSUBN is determined by the type of input sequence. The variable INPUT is then computed to be the sum of XSUBN and the  $a_k$ 's times the delayed INPUT values. The filter output is the sum of the  $b_r$ 's times the delayed INPUT values. These output values are then multiplied by the filter gain, FLTRGN, and stored in the appropriate array: YI for the unit sample response, YS for the unit step response.

### C. FREQUENCY RESPONSE

For discrete systems, the frequency response is found by evaluating the system transfer function at points on the unit circle. Subroutine FRQNCY generates both the phase and the magnitude of the transfer function.

The important variables in the FRQNCY subroutine are: AA, BB, THETA, Z, NNUM, and NDEN. The variables AA and BB are the double precision values of the A and B vectors described in the section on time response. THETA is the angular displacement around the unit circle for which a value of the frequency response is calculated. Z is the





classical z-plane variable:

$$Z = \exp(j \text{ THETA}) \quad (3.3)$$

NNUM is the number of numerator terms. NDEN is the number of denominator terms.

The frequency response is computed for 1024 points around the unit circle. All points computed are for THETA greater than zero and less than  $\pi$ . The values of Z for THETA greater than  $-\pi$  and less than zero are neither computed nor plotted since the magnitude of the transfer function is an even function of THETA and the phase response is an odd function of THETA.

The generation of the  $a_k$  and  $b_r$  coefficients has already been discussed in the section on time domain response. The evaluation of  $H(z)$  is straightforward. The value of THETA is incremented through 1024 steps from zero to  $\pi$ . For each value of THETA generated, the value of Z is computed. The value of  $H(z)$  is then computed in two parts. The numerator is the sum of NNUM terms of the form  $b_r$  times  $z^{-r}$  for  $r=0, 1, 2, \dots, \text{NNUM}-1$ <sup>1</sup>. The denominator is one minus the sum of (NDEN-1) terms of the form  $a_k$  times  $z^{-k}$  for  $k=1, 2, 3, \dots, \text{NDEN}-1$ <sup>2</sup>. Dividing numerator by denominator

---

<sup>1</sup> NNUM is equal to the degree of the numerator plus one.

<sup>2</sup> NDEN equals the degree of the denominator plus one.



yields  $H(z)$ . The phase of  $H(z)$  is then determined using the principal values of the arctangent function; the results are stored in the vector YP. The magnitude of  $H(z)$  is computed, multiplied by FLTRGN, the filter gain, and subsequently stored in the vector YM.

#### D. OUTPUT PLOTS

The output plots generated by ASDF are generated by two different software programs each being different implementations of the same plotting requirements. One set of plots is generated using PLOT-10 software on the Tektronix 4012 terminal. Another set of plots utilizes the Versatec plotter and associated software. [Ref. 7] The PLOT-10 software is adequately covered in reference 10. The Versatec software is explained in Technical Note-034 in the computer center.

The plots which are generated on both devices are essentially the same. There are five plots generated on the 4012 terminal: unit sample response, unit step response, phase response, magnitude of the transfer function, and magnitude of the transfer function in decibels. Both of the time domain plots are terminated when the output value has stabilized to within one percent of the maximum generated system output. All 1024 points of the phase and magnitude plots are plotted.



The form of the Versatec output is similar to that of the 4012. One additional plot is included. It shows the unit circle with poles and zeros of the filter being analyzed. Plotting is terminated when the filter output stabilizes to within one percent of the maximum system output generated.

On both devices, the values of unit sample response are stored in a vector YI and plotted with subroutine PLTIMP. The unit step response values are stored in YS, and plotted with subroutine PLTSTP. The phase response is stored in vector YP. The magnitude of the transfer function is stored in vector YM. The magnitude of the transfer function in decibels is stored in YMDB. The three vectors YP, YM, and YMDB are all plotted with subroutine PLTTRE. The Versatec plot of the unit circle is generated with subroutine UNTCR. The 4012 plot of the unit circle and command array are generated with the subroutine UNTCIR.

#### E. SUPERVISORY ROUTINES

There are three supervisory routines used with the ASDF programs. These supervisory routines make the loading of ASDF transparent to the user. There is a substantial amount of file manipulation required to store and load the ASDF programs. A knowledge of the hardware and software available for these purposes is not particularly useful to the user. For these reasons, the supervisory routines are discussed only briefly. Two of these routines are based on





CP/CMS and IBM EXEC environment; the third is written in the IBM-360 Job Control Language.

The JCL program moves the ASDF program from its permanent storage location on DISK02 to the CP/CMS disks OSX001. The two EXEC routines generate an appropriate CP/CMS work space, and manipulate the ASDF text file into a usable form.

The EXEC program ASDF191 resides on the user 191-P-disk. This routine automatically requests and formats ten cylinders of temporary disk space. The ASDF TEXT file is moved from the OSX001 disk to the newly created temporary disk space 192-T. Once in place, The ASDF TEXT file name is changed to ASDF EXEC to generate the next file with the correct file type. ASDF EXEC is then edited to remove the first set of data cards which hold the ASDF192 EXEC program. The 192-T-disk is released and the user is requested to type two commands to the system. The "login 192 P" command makes the 192 disk the P-disk. The "EXEC ASDF192" command begins the execution of the second EXEC file.

The general ASDF192 EXEC structure is depicted in figure 3-6. ASDF192 EXEC is created by ASDF191 EXEC and resides on disk 192. In a manner similar to ASDF191, ASDF192 splits the ASDF TEXT file, which contains all of the component parts of the ASDF program, into a series of smaller files. Once all files have been separated, titles are altered appropriately, text files are loaded into core, and load



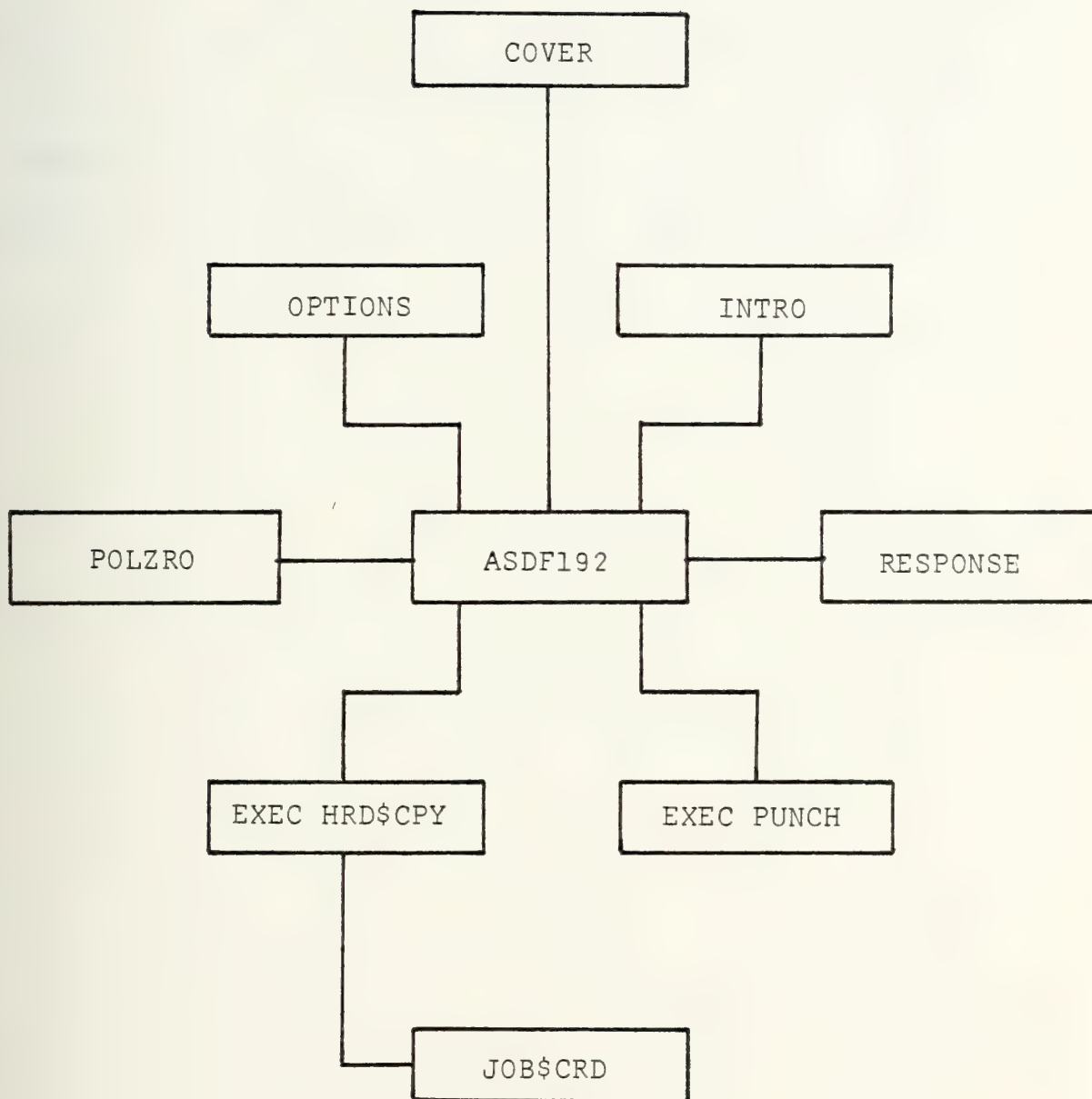


Figure 3-6 Structure Of ASDF192 EXEC



modules are generated. ASDF192 then reverts to an infinite loop, reading and implementing the desired user command. There are six user commands available to the user: POLZRO, RESPONSE, EXEC PUNCH, EXEC HRD\$CPY, OPTIONS, and INTRO. The applicability of these commands is briefly discussed in chapter four. This brief description is also presented by executing the OPTIONS command once ASDF has been loaded.





#### IV. STRUCTURE OF ASDF COMMANDS

The ASDF program generates the time and frequency responses of realizable digital filters of order ten or less. Chapter three describes the algorithms used for these computations. The procedure for executing the ASDF program generally consists of two distinct steps. The first step is to enter the poles and zeros of the filter to be analyzed. The second step is to perform the calculations required to generate and plot the desired filter responses. There are six commands which may be executed within the ASDF environment providing the user with the ability to control the form of the output, to enter the poles and zeros of the filter, and to clarify most questions pertaining to the meaning and use of the general commands. The six fundamental commands available to the user are: OPTIONS, INTRO, POLZRO, RESPONSE, EXEC PUNCH, and EXEC HRD\$COPY. Only the POLZRO command has a subordinate command structure. All of these commands are issued by typing the command on the terminal keyboard in response to: \*ASDF-RE.

The OPTIONS command provides a listing of all commands available to the user. Each command is accompanied by a brief description of its intended use.

The INTRO command writes on the screen six pages of information pertaining to the POLZRO command substructure.



INTRO includes a general description of the data input modes, and a description of several input situations with the appropriate executable commands.

The POLZRO command allows the user to modify or create a filter in the z-plane. All adjustment of poles and zeros must take place with this command. By issuing this command the user may add and delete poles and/or zeros as well as adjust the filter gain constant. The commands which are issued within the POLZRO environment are described in a later section of this chapter.

The RESPONSE command displays the characteristics of the filter being created by the user. Five displays are presented: unit sample response, unit step response, phase of the transfer function, magnitude of the transfer function, and the magnitude of the transfer function in decibels.

The EXEC PUNCH command allows the user to punch a card deck of the filter pole and zero locations on the offline punch. This punched output can be picked up in room I-140 of the computer center. The cards punched by this command may be subsequently reentered into the system so that this particular filter can be referenced or modified at a later date.

When the user desires to enter a filter in this way, the punched cards are presented to one of the computer operators over the counter in room I-140. The cards are punched in



the correct format for direct submission into the CP/CMS card reader. The operator subsequently reads this deck into the user's virtual card reader. As the user signs onto a terminal and gains access to CP/CMS, the computer includes an additional message in the standard login typeout showing that the user has a virtual reader file. This message is of the form:

FILES:- 01 RDR, NO PRT, NO PUN

When the user wishes to access this old filter data he must execute the CP/CMS command OFFLINE READ \* \*. The OFFLINE READ command will read the punched card filter data from the user's virtual card reader, and generate a file on the user's disk space entitled FILE FT01F001. Any files with this name present before the OFFLINE READ command is issued will be over-written by the new file. If the user does not wish to lose an older FILE FT01F001, he should use the CP/CMS ALTER command to change the name of the old file that he wishes to protect. Once the user has successfully loaded the ASDF program, the ASDF signature i.e., \*ASDF-RE, is presented. The OFFLINE READ command may be issued any time that this signature appears.

The EXEC HRD\$CPY command is available only when the SUBMIT function of CP/CMS is functioning. When the SUBMIT command is not functional a message is sent to the user during the login procedure which states that SUBMIT is not available. the output from this command is ultimately





available in the computer center, and consists of two parts. The first part of the output is the numerical listing of the values of the responses computed by the ASDF program. The EXEC HRD\$CPY command is the only command which will allow the user access to these numerical values. The second part of the output is a series of Versatec plots, each similar to the screen presentations of the RESPONSE command, with one additional plot of the unit circle and pole/zero locations. When the EXEC HRD\$CPY command is executed the ASDF program will ask several questions, using the responses to generate a JOB card. The response to the "ENTER DESIRED JOB NAME" is the title which appears on the final output. ASDF then submits the filter characteristics to the operating system for processing. The hard copy output will be found in the standard output boxes.

#### A. SUMMARY OF DATA INPUT MODES

There are three modes through which the poles and zeros may be entered into the z-plane: 1) interactive graphics, 2) rectangular, and 3) polar. Each of these modes has particular attributes and limitations.

##### 1. Interactive Graphics (IAG)

This mode was designed to enter poles and zeros rapidly, in cases where accuracy is not of prime importance. The interactive graphics mode is characterized by the appearance of the graphics cursor (the intersection of two



lines which trace across the terminal screen). The position of the cursor is controlled by the two thumb wheels located to the right of the interactive ASCII keyboard on the Tektronix 4012 terminal. The coverage of the interactive graphics cursor is limited to values of X and Y greater than -1.1 and less than +1.1. Zeros outside this area must be entered with either the polar or rectangular commands. Careful adjustment of the cursor can usually locate poles or zeros within .002 units of the desired location. Poles must be located within the unit circle.

## 2. Rectangular

This mode was designed to enter from the terminal keyboard, poles and zeros which are known in rectangular coordinates. It is slower than the interactive graphics mode because there is a significant amount of screen rewriting required. The rectangular mode allows much greater accuracy in the location of poles and zeros. Poles may not be entered on or outside the unit circle. Zeros must be located within ten units of the z-plane origin.

## 3. Polar

This mode was designed to enter from the terminal keyboard, poles and zeros which are known in polar coordinates. This mode is also slower than the interactive graphics mode. It allows poles and zeros to be entered to any permissible position in the z-plane. Poles must be entered within the unit circle. Zeros must be entered within



ten units of the z-plane origin. Each polar command has a description of the constraints on the data to be entered automatically written on the screen. This mode is particularly appropriate when constructing all-pass linear phase filters.

## B. THE POLZRO COMMAND SUBSTRUCTURE

In an attempt to maximize execution speed, a minimum of information is written on the screen, thus reducing the rewrite time after execution of each command. When program execution begins, the user is presented with the unit circle, and a listing of commands on the top left side of the screen. The commands are abbreviated in the following manner. The general command type or mode is specified across the top of the array as INTERACTIVE, RECT, and POLAR (see figure 4-1). Each of these modes conforms to the above description. Down the left side of the array are the commands of the form CRR, where C is the change to be made, A - for add a root, D - for delete a root. RR is for the type of root to be acted upon by the command: RZ - real zero, CZ - complex zero, RP - real pole, and CP - complex pole. As an example, the box in the center column aligned with the row titled DCZ stands for the command "Delete a Complex Zero - Rectangular Coordinates. There are two unique commands listed as NUC, and LST. The New Unit Circle (NUC) command simply redraws the unit circle and command



| UNIT CIRCLE COMMANDS |                          |                          |                          |                          |
|----------------------|--------------------------|--------------------------|--------------------------|--------------------------|
|                      | INTER-<br>ACTIVE         | KEYBOARD<br>RECT         | POLAR                    |                          |
| CMDS                 |                          |                          |                          |                          |
| ACZ                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| ACP                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| ARZ                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| ARP                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DCZ                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DCP                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DRZ                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DRP                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| NUC                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| LST                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

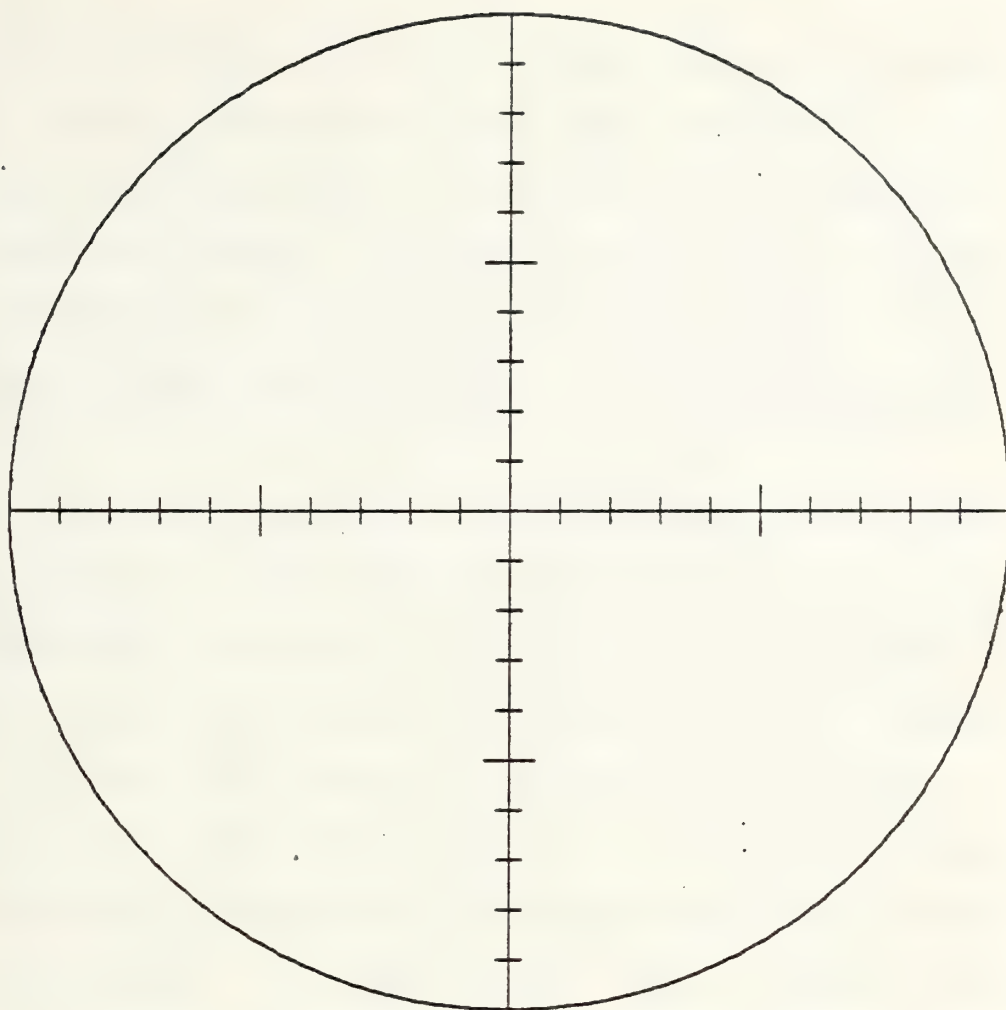


Figure 4-1 Basic Display Format





array. This is appropriate when either too many entries have been made to the terminal, and the command array is no longer clear, or when noise in the line from the computer causes a deformed array or unit circle to be drawn. The LST command provides the user with a listing of the current status of the filter being analyzed, in a self explanatory format.

In all cases when the unit circle and command array just described appear on the screen, an audible tone or bell is heard. This bell indicates to the user that the cursor is to be positioned within one of the boxes in the command array. In order to execute a command, the cursor is positioned in the box associated with the desired command. A single keyboard character (any character but the "return" key is acceptable) is then typed. The cursor then disappears. The system presents a carat pointing to the command that the computer is about to execute; the flashing cursor blinks within the box associated with the command which is about to be executed. If the box so indicated is the command that was selected, the user types a "space", an "e" (for execute) and then "carriage return". In cases where a rectangular or polar mode command is to be executed, the screen is then erased, and new information is printed. In cases where an interactive graphics command is to be executed, only the cursor reappears. This second appearance of the cursor is not accompanied by a bell or audible tone.



This indicates that the cursor is to be positioned within the unit circle. Once so positioned, a single keyboard character is typed (not "return"), and, as appropriate, the unit circle is redrawn to reflect the change in pole or zero locations. After this update of the z-plane, the circle and command array are redrawn with the cursor accompanied by the audible tone.

There are only two exceptions to this scenario. First, if the flashing cursor does not flash in the correct box, the wrong command will be executed. If an incorrect box contains the flashing cursor, the user should not type the "space", "e", sequence. Instead, he should type two other characters (e.g., "space", "space") and then "return." The cursor returns along with the audible tone, calling for a readjustment of the cursor, closer to the desired command box. The second exception occurs when the placement of all of the poles and zeros is completed. In this case, after aligning the cursor with any box, the sequence "space", "x" (for exit), "return" should be typed. This will terminate the input, or modification, stage of the program and present the user with a series of options for the next stage of filter processing. At this point a display appears which explains the options available to the user.



## C. SUMMARY OF POLZRO SUBCOMMANDS

When the cursor appears along with an audible tone, the user is to position the cursor within the command array, and type a keyboard character (not "return"). When the flashing cursor appears within a box, he types ("space", "e", "return") to execute the specified command. "Space" and any other character but "x" will return the user to the command table for cursor readjustment. To terminate the input, or modification mode, the cursor is positioned over any box, and once the flashing cursor enters the box, the characters "space", "x", "return" are typed.





## V. LOADING THE ASDF PACKAGE

The procedure for using the ASDF program consists of either six or seven steps depending on whether or not the ASDF package is resident on OSX001. Usually the program sections of ASDF are stored on the T2314 disk pack OSX001. Each Monday morning the OSX001 disks are scratched so that the ASDF program must be moved from the permanent storage location on DISK02, to the temporary storage location on OSX001. The user may simply sign onto a 4012 terminal, and execute three commands to load ASDF into his virtual machine. Once this loading process has been successfully completed, the commands previously discussed are at the user's disposal for filter analysis.

### A. LOGIN PROCEDURE

With one minor exception, the standard login procedures (specified in Section 4.4.3 of the CP/CMS User's Manual) [Ref. 4] are applicable when entering the CP/CMS system to use ASDF. Due to the size of the ASDF programs the user must request a 400k virtual machine when logging onto CP/CMS. The virtual machine is requested as follows:

```
login XXXXTNN 400k
```

where XXXX is the user's computer center number, T is the type of user i.e. G for general user, and P for private



user, and NN is the two digit terminal number. The 400k request in the LOGIN is absolutely mandatory. Without this request the user will be able to load the ASDF program, however, when attempting to execute the RESPONSE command, an error: "INPUT COMMAND ERROR" will result. Should this error occur when the command is correctly spelled the user has no alternative but to logoff and log back on to the system requesting the correct size virtual machine. The remainder of the login procedure is identical to the examples in the CP/CMS user's manual.

#### B. TESTING OSX001 FOR ASDF PROGRAMS

The second step in loading the ASDF programs is to determine whether or not a copy of the programs exists on OSX001. The ASDF programs are permanently stored on the operating system DISK02. The first time that the programs are used each week, the user must move them from DISK02 to the disk OSX001. To determine whether or not OSX001 has the ASDF package, issue the command:

```
TOCP ASDF191 EXEC P1
```

The system will respond:

```
EXECUTION BEGINS...
```

```
237 = OSX001
```

```
ENTER DSNAME OR '/'
```

The correct user response is:

```
F0718.asdf191
```



At this time one of two responses will appear. The particular response received is dependent on whether or not ASDF resides on OSX001.

1. ASDF is on OSX001

When the ASDF programs are located on OSX001, the system response is:

EXTENTS EXHAUSTED, EOF SIMULATED.

4 BLOCKS 63 RECORDS READ

This message indicates that file ASDF191 has satisfactorially been transferred to the user's virtual machine and the user is ready to continue with the subsequent loading steps.

2. ASDF is not on OSX001

When the ASDF programs are not on OSX001, the system response will be:

DATA SET NOT FOUND (OS/360 S213)

ENTER DSNAME OR '/'

This response indicates that the user must move the ASDF programs to OSX001 before the loading procedure may be resumed. The simplest method for accomplishing this transfer is to generate a file on CP/CMS which exactly duplicates the Job Control Language program which follows.

To generate the file which transfers ASDF, logon to CP/CMS. Once the system's responses have been entered and the system is ready, issue the command:

EDIT ASDF JCL



Then type in, or read in<sup>1</sup>, the program which follows.

```
// Standard JOB Card (User's Manual Sec. 3.3.2.1)
// EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSIN DD DUMMY
//SYSUT1 DD DSN=F0718.ASDF,VOL=SER=DISK02,UNIT=3330,
//   DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200),DISP=(OLD,KEEP)
//SYSUT2 DD DSN=F0718.ASDF,VOL=SER=OSX001,UNIT=2314,
//   DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200),DISP=(NEW,KEEP),
//   SPACE=(TRK,(40,1),RLSE)
// EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSIN DD DUMMY
//SYSUT1 DD DSN=F0718.ASDF191,VOL=SER=DISK02,UNIT=3330,
//   DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),DISP=(OLD,KEEP)
//SYSUT2 DD DSN=F0718.ASDF191,VOL=SER=OSX001,UNIT=2314,
//   DCB=(RECFM=FB,LRECL=80,BLKSIZE=-800),DISP=NEW,KEEP),
//   SPACE=(800,(3,1),RLSE)
//
```

Once this program has been correctly read in to the user's disk, it must be submitted to the operating system using the standard SUBMIT command:

```
SUBMIT ASDF JCL P1 XXXX
```

XXXX is the user's computer center number. If the SUBMIT function is working the following messages will be returned:

```
** CARDS XFERED TO 0098P **
```

```
FNAME FTYPE P1 SENT FOR OS BATCH JOB
```

After a short delay the system will send a second message:

```
FROM 0098P33 : JOB SENT TO OS WITH 21 CARDS WRITTEN (Time)
```

The user should then revert back to section B above which

---

<sup>1</sup>A deck containing this program is available from Professor Kirk, Sp 302.





discusses the TOCP command. After giving the system a period of time to execute the JCL job which was just submitted, execution of the TOCP command will indicate that the ASDF files are present. If the operating system has not completed executing the transfer, the TOCP request will respond as in paragraph two above.

## B. LOADING ASDF

Once ASDF resides on disk OSX001, the remaining required steps are simple. The user issues the command:

```
EXEC ASDF191
```

The system will type a series of lines onto the terminal. At the conclusion of these typewritten lines the user is prompted to issue the next two commands:

```
LOGIN 192 P
```

The system responds:

```
192 REPLACES P (191)
```

The next required entry is:

```
EXEC ASDF192
```

These two commands will cause a very long list of CP/CMS commands to be typed on the screen. Each of these CP/CMS commands is automatically executed by the operating system in the process of loading the ASDF programs. There are no further commands required from the user to complete the loading process.



When the entire ASDF program package is loaded, the user will be presented with a title page and the instruction: "HIT SPACE AND RETURN TO CONTINUE." Once the user has typed "space" and "return" the display will write out a listing of the commands at his disposal. These commands have already been discussed in chapter four.



## VI. USING ASDF

Each of the following sections constitutes an example for one of the three input modes. Each example uses only one input mode. In general, these modes may be combined during the generation of any particular filter. The mode selected should be the one most convenient to the user.

### A. EXAMPLE ONE - INTERACTIVE GRAPHICS

This first example uses the ASDF program in the interactive graphics mode. The objective is to observe the responses of a digital filter which has two real zeros, one at the origin and another at +0.8, two complex poles located at  $(0.8 \pm j0.2)$ , and the filter gain is 1.000. The interactive graphics mode is used since the exact locations of the poles and zeros are not critical. For the purposes of this example, it is assumed that the user has already followed the login procedure described in the above paragraphs, and has loaded the ASDF program. The user has already seen the title page, and the two pages of command descriptions. The screen is clear except for the top left corner which shows:

\*ASDF-RE

This group of letters is the signature for the ASDF program





and indicates that the program is ready to execute the user's next command.

Since we must first enter the poles and zeros into the z-plane the appropriate command to type is:

POLZRO

The ASDF program will then draw the screen as shown in figure 6-1, and generate an audible tone. Initially the cursor will be located at some arbitrary position or perhaps may not appear on the screen at all. Movement of the thumb wheels moves the cursor position. Since we wish to execute in the interactive graphics mode, to add a real zero, we move the cursor (intersection of the two lines tracing across the screen) into the box in the left column labeled ARZ, (Add a Real Zero) figure 6-2. We then type any keyboard character except "return".<sup>1</sup> The trace cursor will disappear, and the flashing alphanumeric cursor will appear. The program will print a carat pointing to the box selected by the user. The flashing cursor will move into the box selected.

The user then types the command "space", "e", "return". The flashing cursor disappears, and the trace cursor reappears. The absence of an audible tone at this time indicates that the trace cursor is to be positioned within the unit circle.

---

<sup>1</sup>The 4012 terminal in the computer center requires a "control-s" after every response.



# UNIT CIRCLE COMMANDS

| CMDS | INTER-                   |                          | KEYBOARD                 |                          |
|------|--------------------------|--------------------------|--------------------------|--------------------------|
|      | ACTIVE                   | RECT                     | POLAR                    |                          |
| ACZ  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| ACP  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| ARZ  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| ARP  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DCZ  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DCP  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DRZ  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DRP  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| NUC  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| LST  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

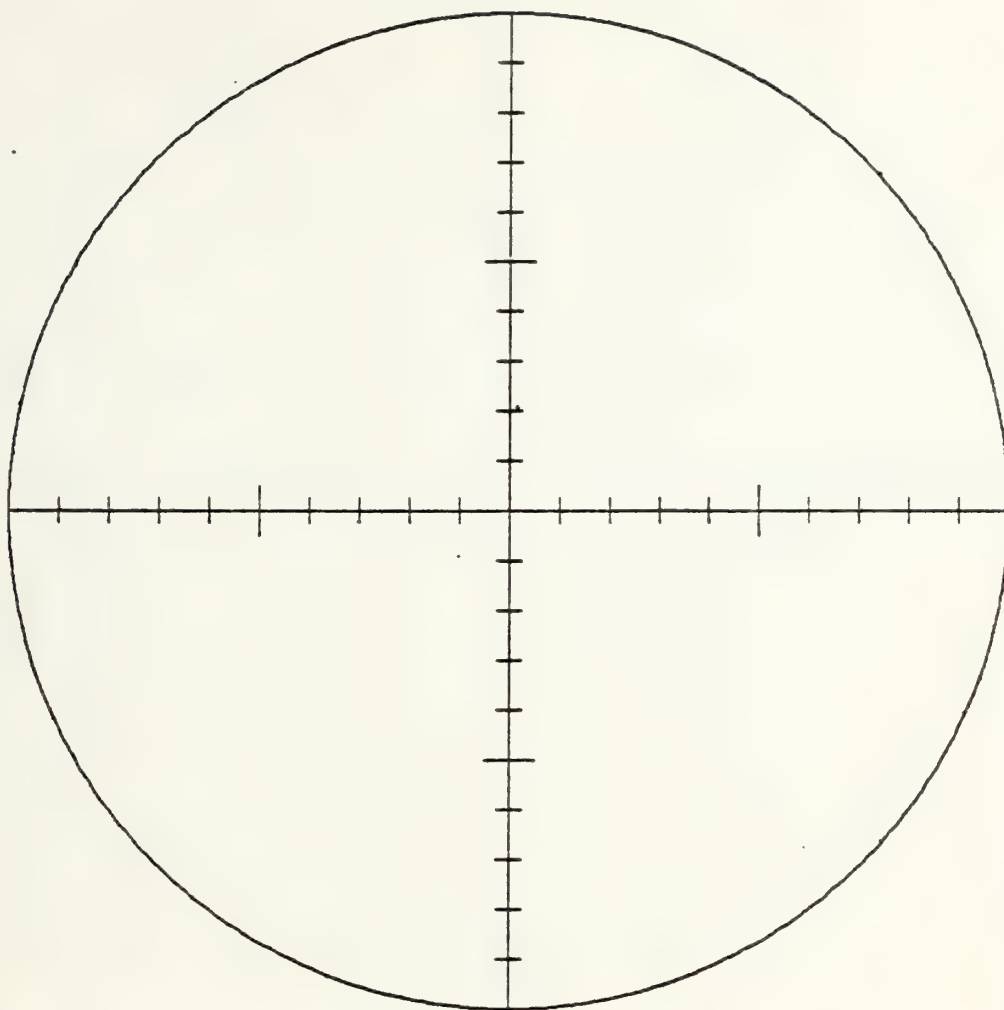


Figure 6-1 Initial Screen Representation After Issuing The Command POLZRO



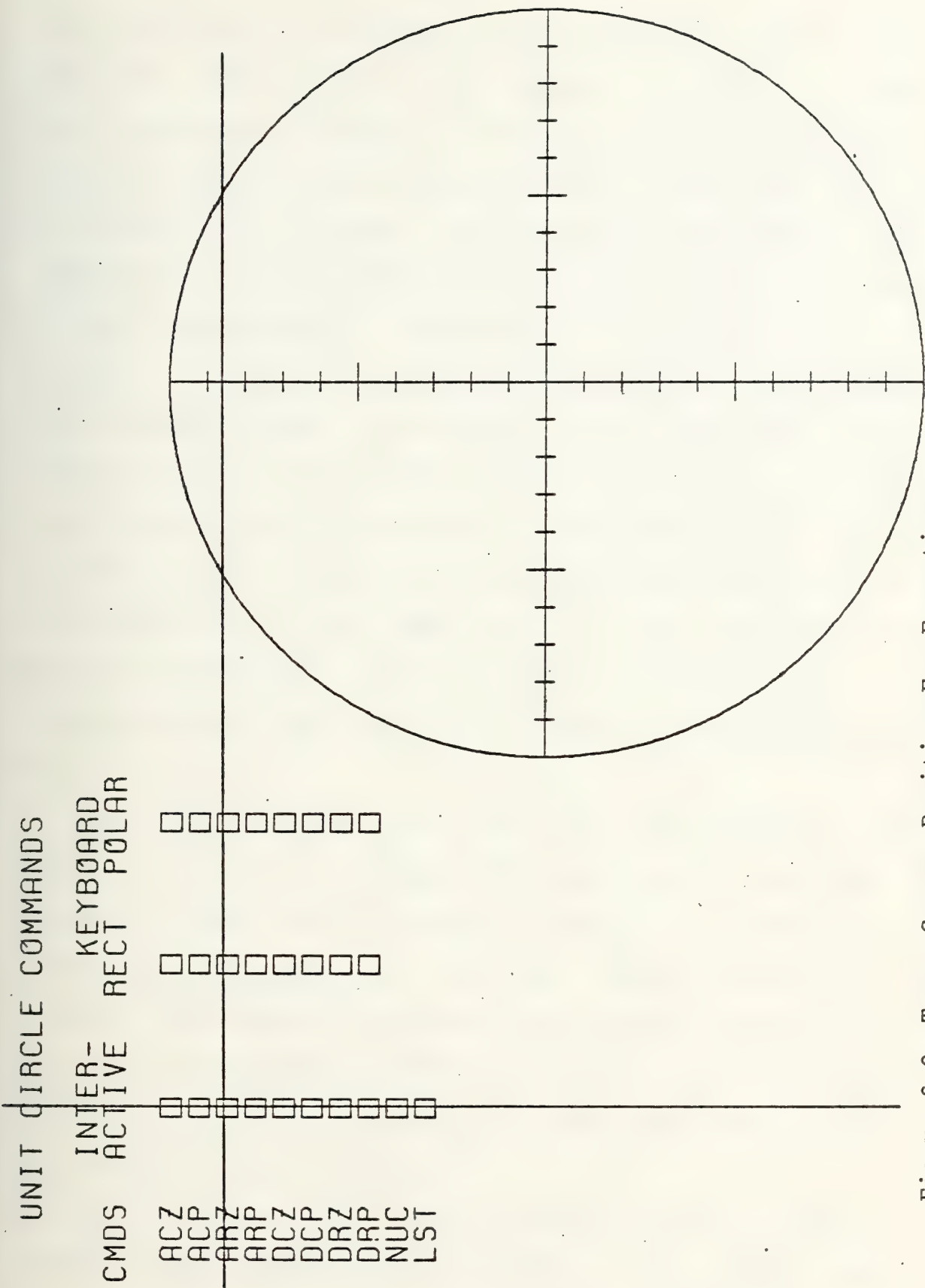


Figure 6-2 Trace Cursor Position For Executing  
The Command "Add A Real Zero"-Interactive Graphics Mode



The trace cursor is adjusted to the position of the desired real zero, (the origin in this case), as shown in figure 6-3. Any keyboard character except "return" is then typed. The ASDF program plots the zero at the origin, and returns the trace cursor with the audible tone. The audible tone reminds the user that the trace cursor is to be positioned in the command array. Since another real zero is to be entered, the cursor is positioned as in figure 6-1b again, and the same steps are followed. When the trace cursor reappears without the audible tone, position it within the unit circle as near as possible to the point  $(0.8 \pm j0.0)$  as shown in figure 6-4. (To enter the zero precisely at 0.8 the rectangular or polar mode would be used.) The steps for entering the complex pole pair exactly parallel those for the previous entries, and are depicted in figures 6-5, and 6-6.

Once all the poles and zeros have been entered, the trace cursor may be aligned with any command box, and a character typed (not "return"). Once the alphanumeric cursor enters that box, the user types "space", "x", "return". The screen is cleared and the user is asked if he wishes to change the default filter gain. Since the default value is 1.0, the user responds "n", the ASDF program clears and responds with: \*ASDF-RE

The next step in the normal sequence is for the user to look at the filter characteristics. To display these





| UNIT CIRCLE COMMANDS |                          |                          |                          |
|----------------------|--------------------------|--------------------------|--------------------------|
| CMDS                 | INTER-<br>ACTIVE         | KEYBOARD<br>RECT         | POLAR                    |
| ACZ                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| ACP                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| ARZ                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| ARP                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DCZ                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DCP                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DRZ                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DRP                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| NUC                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| LST                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

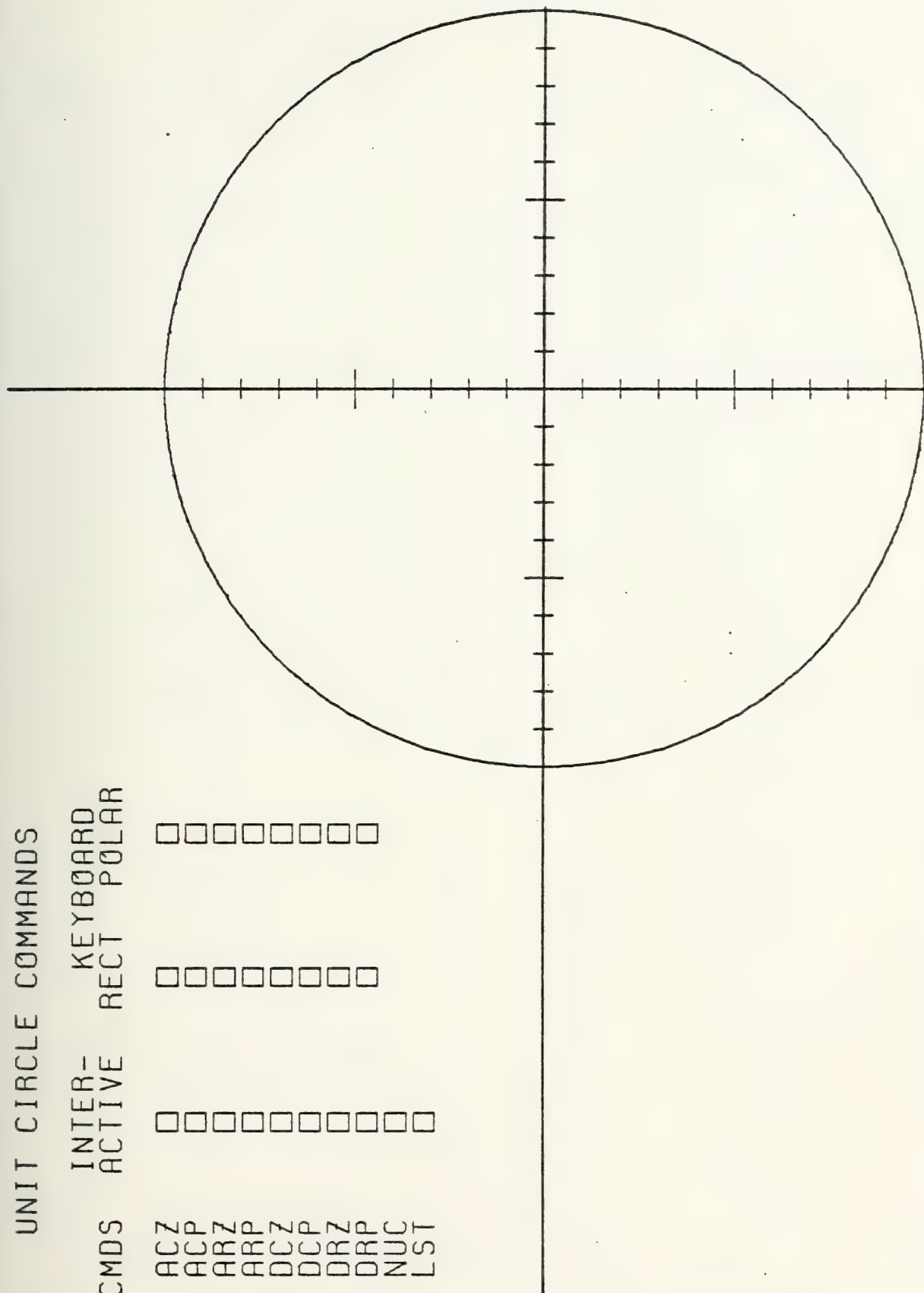


Figure 6-3 Trace Cursor Position Required To Place A Real Zero At The Origin



# UNIT CIRCLE COMMANDS

| CMDs | INTER-                   | KEYBOARD                 |                          |
|------|--------------------------|--------------------------|--------------------------|
|      | ACTIVE                   | RECT                     | POLAR                    |
| ACZ  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| ACP  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| ARZ  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| ARP  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DCZ  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DCP  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DRZ  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DRP  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| NUC  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| LST  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐  
☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐  
☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐

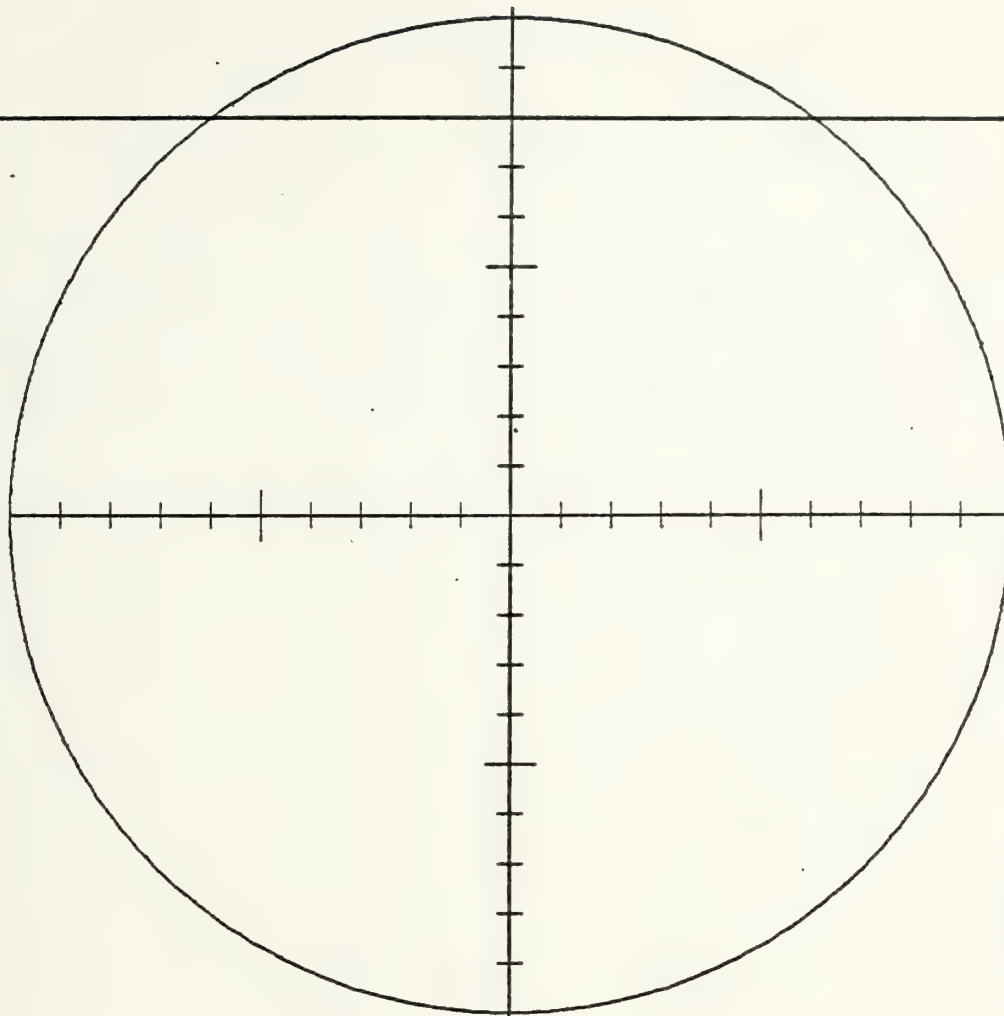
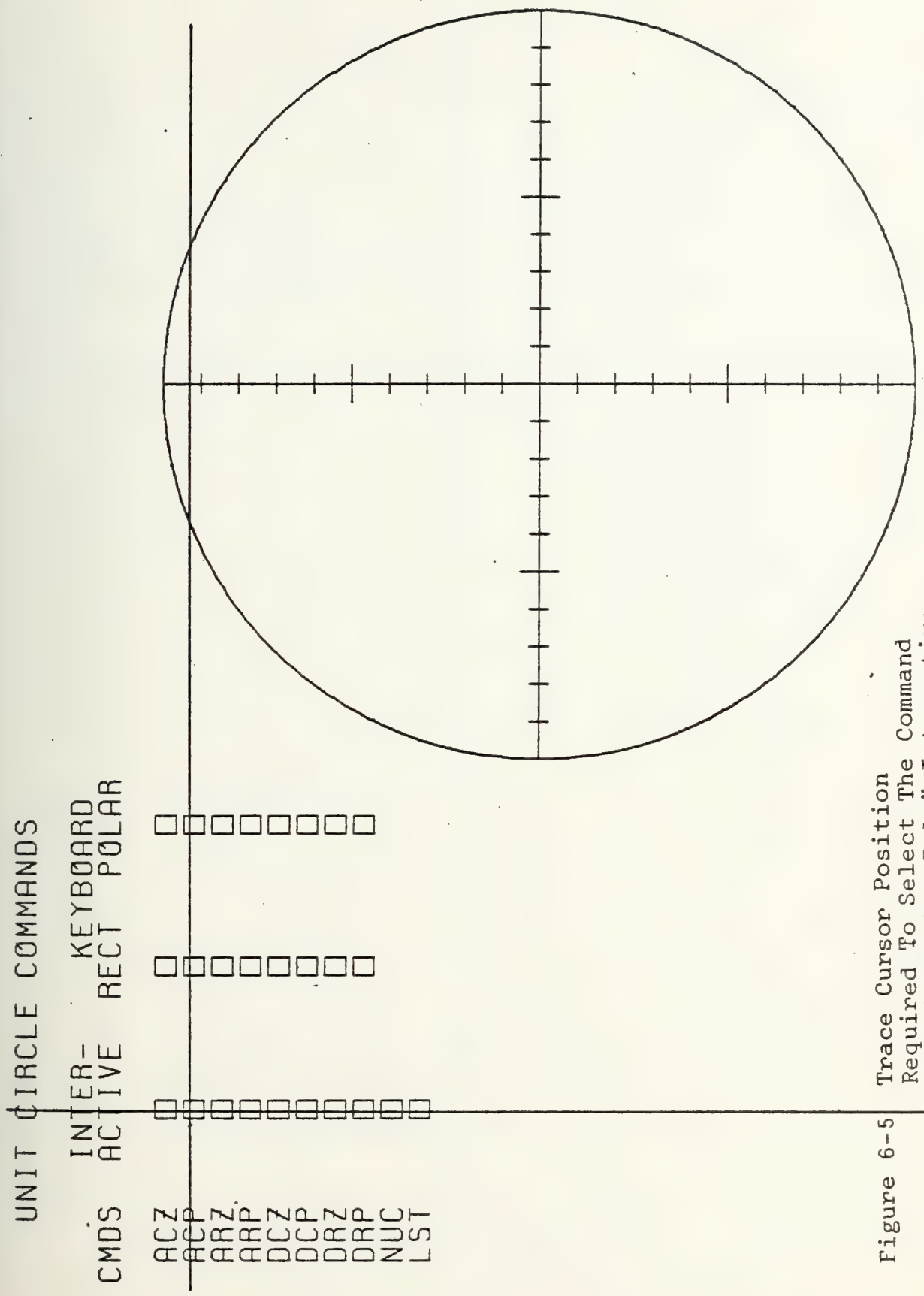


Figure 6-4 Trace Cursor Position Required To Put A Real Zero At (+0.8)





Trace Cursor Position  
Required To Select The Command  
"Add A Complex Pole"-Interactive  
Graphics Mode

Figure 6-5





# UNIT CIRCLE COMMANDS

CMDS INTER- KEYBOARD  
ACTIVE RECT POLAR

|     |                          |                          |                          |
|-----|--------------------------|--------------------------|--------------------------|
| ACZ | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| ACP | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| ARZ | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| ARP | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DCZ | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DCP | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DRZ | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DRP | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| NUC | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| LST | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

> ☐ e

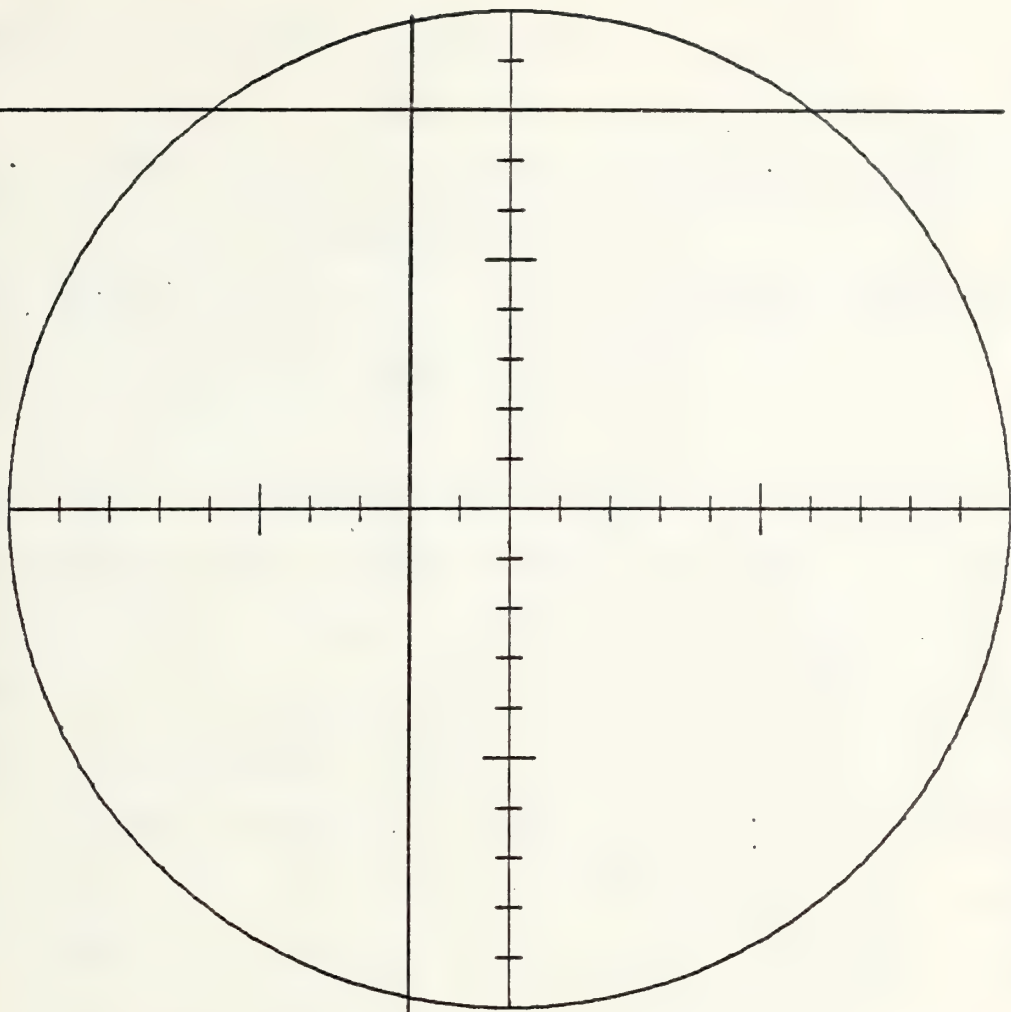


Figure 6-6 Trace Cursor Position Required To Enter A Complex Pole At (0.8+/-j0.2)



characteristics the command typed is:

#### RESPONSE

To get a hard copy of the filter responses the command:

EXEC HRD\$CPY

is typed. Figures 6-7a, b, c, d, e, f, and g show the pertinent portions of the output.

### B. EXAMPLE TWO - RECTANGULAR MODE

This second filter example is taken from [Ref. 5], page 222. The filter is a fourth-order low pass Chebyshev. The derivation of the transfer function is described in the text. The given transfer function is factored to give the desired pole/zero locations. The filter has two complex zeros at  $(-1.0 \pm j0.0)$  and two complex pole pairs, one pair at  $(.7498 \pm j.5348)$ , and one pair at  $(.7774 \pm j.2120)$ . The filter gain is given as 0.001836.

In a manner exactly paralleling the previous example, the user selects the command ACZRCT (which stands for Add a Complex Zero - Rectangular Coordinates), by positioning the trace cursor in the appropriate box, as in figure 6-8a. Once a keyboard character is typed, and the alphanumeric cursor moves into the box, the "space", "e", and "return" keys are typed. The ASDF program will erase the screen and draw the display shown in figure 6-8b. The user then types in the rectangular coordinates of either complex zero in the first complex zero pair, figure 6-8c. The system will



```

NUMBER OF REAL ZEROES: 2 PAIRS: 0
NUMBER OF COMPLEX ZEROES: 0
NUMBER OF REAL POLES: 0
NUMBER OF COMPLEX POLES: 1
FILTER GAIN: 1.0000000

```

Figure 6-7a Pole Zero Locations For Example Two



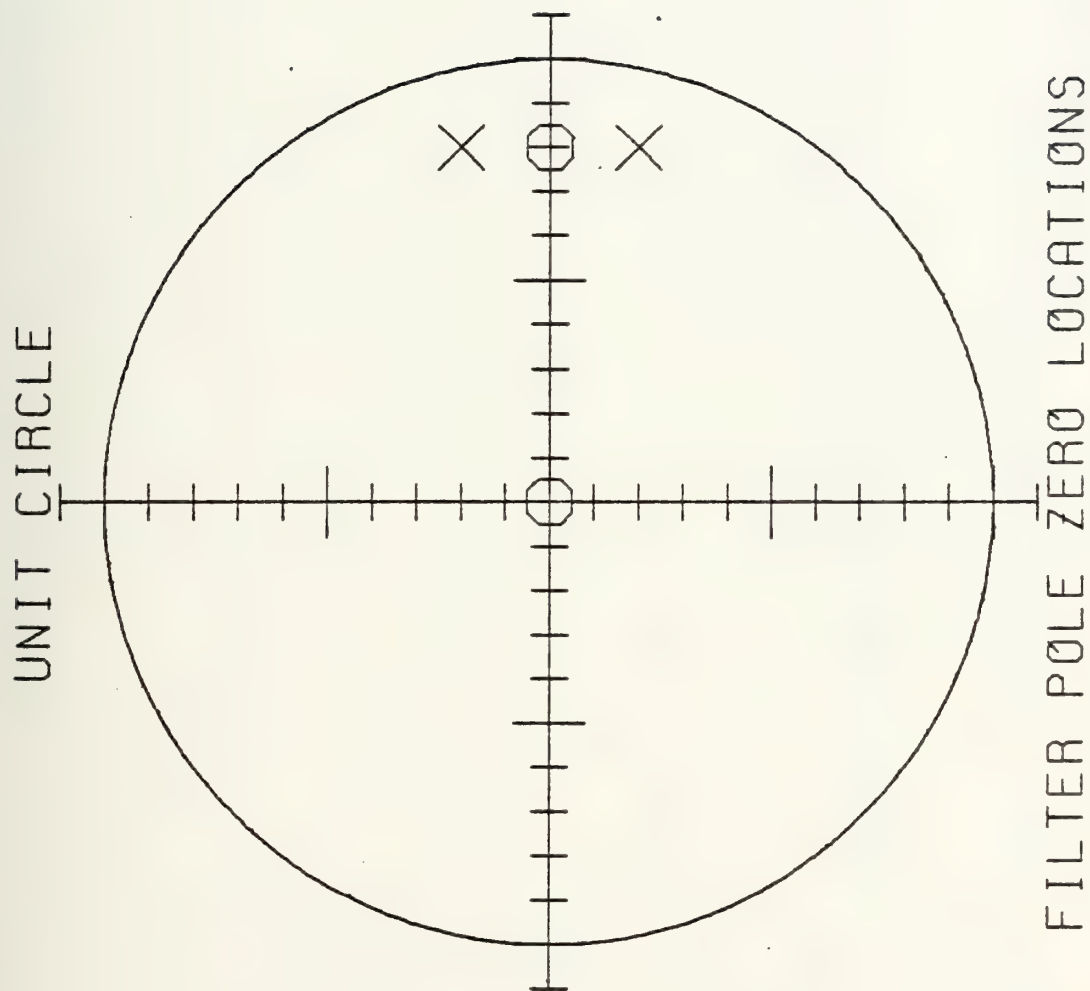


Figure 6-7b Pole Zero Locations For Example Two





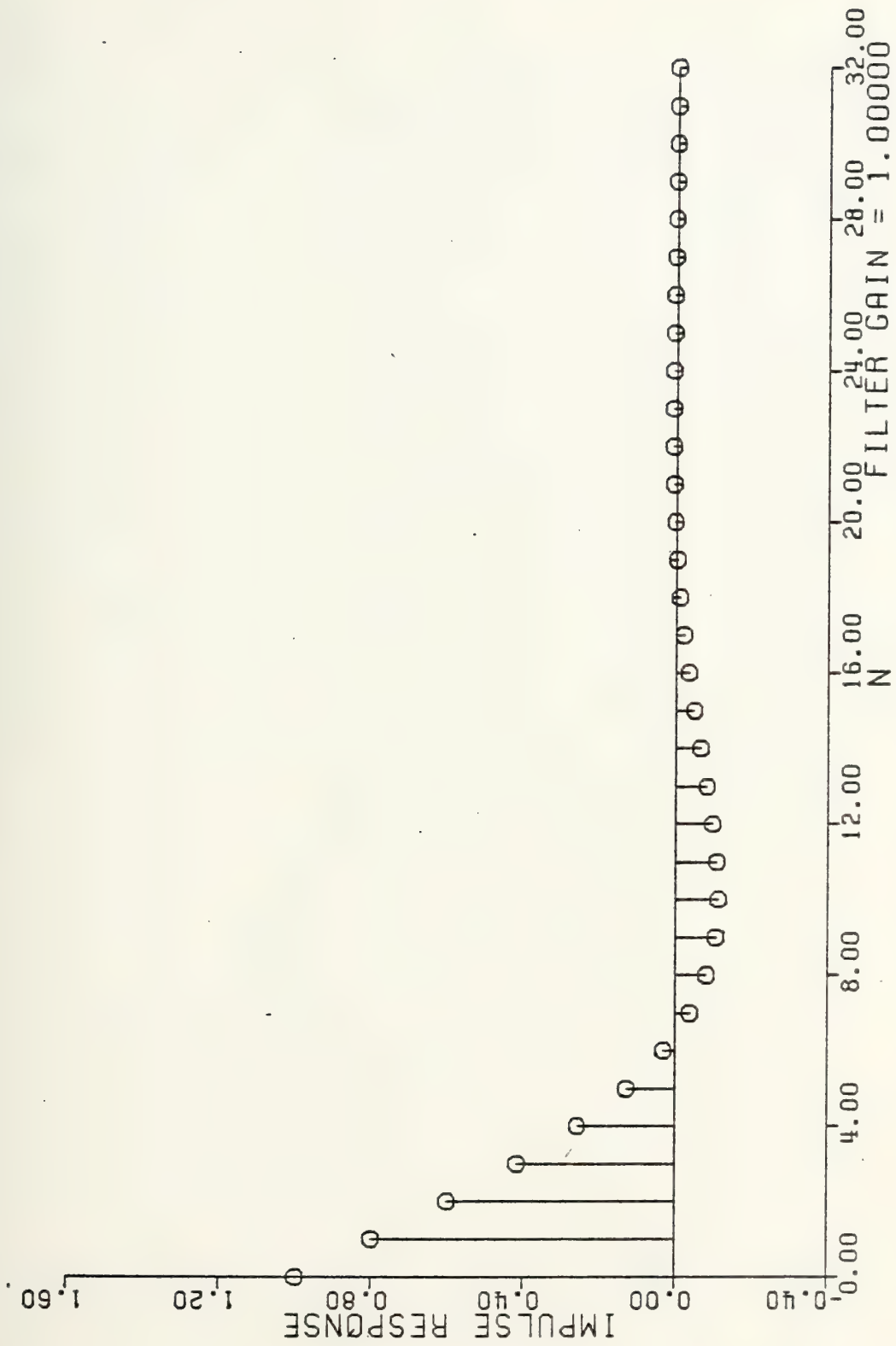


Figure 6-7c Unit Sample Response For Example Two



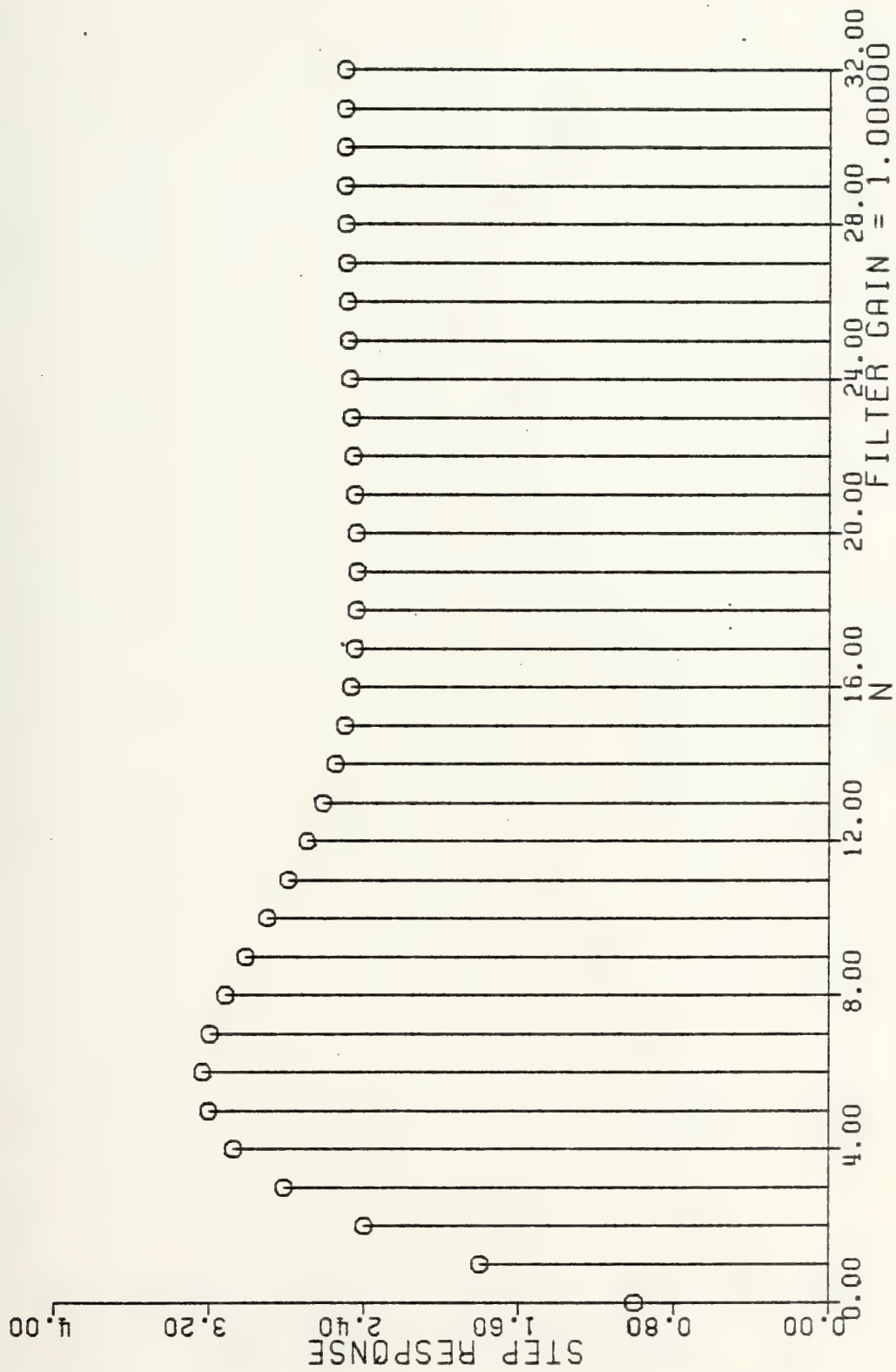


Figure 6-7d Step Response For Example Two



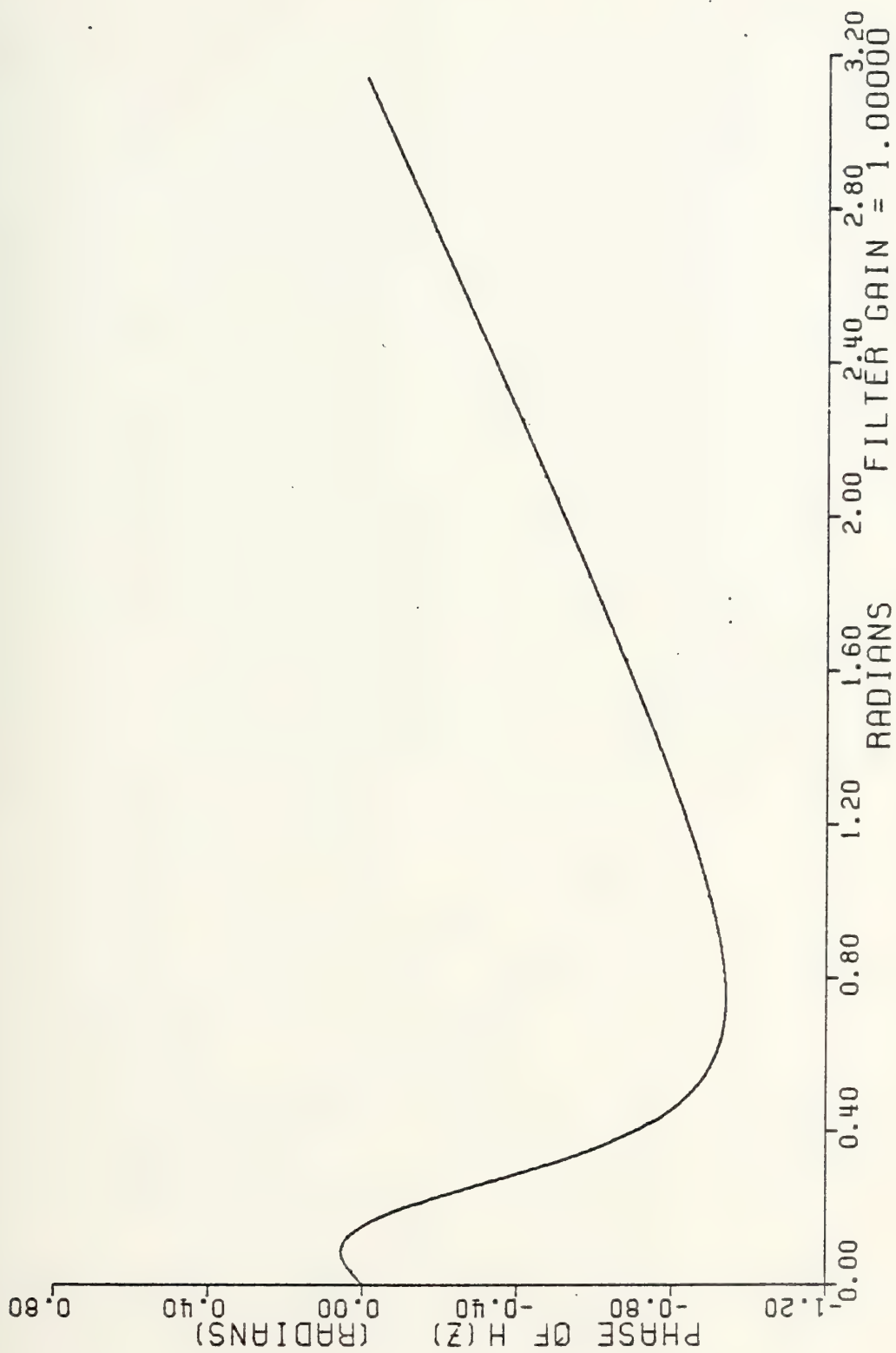


Figure 6-7e Phase Response For Example Two



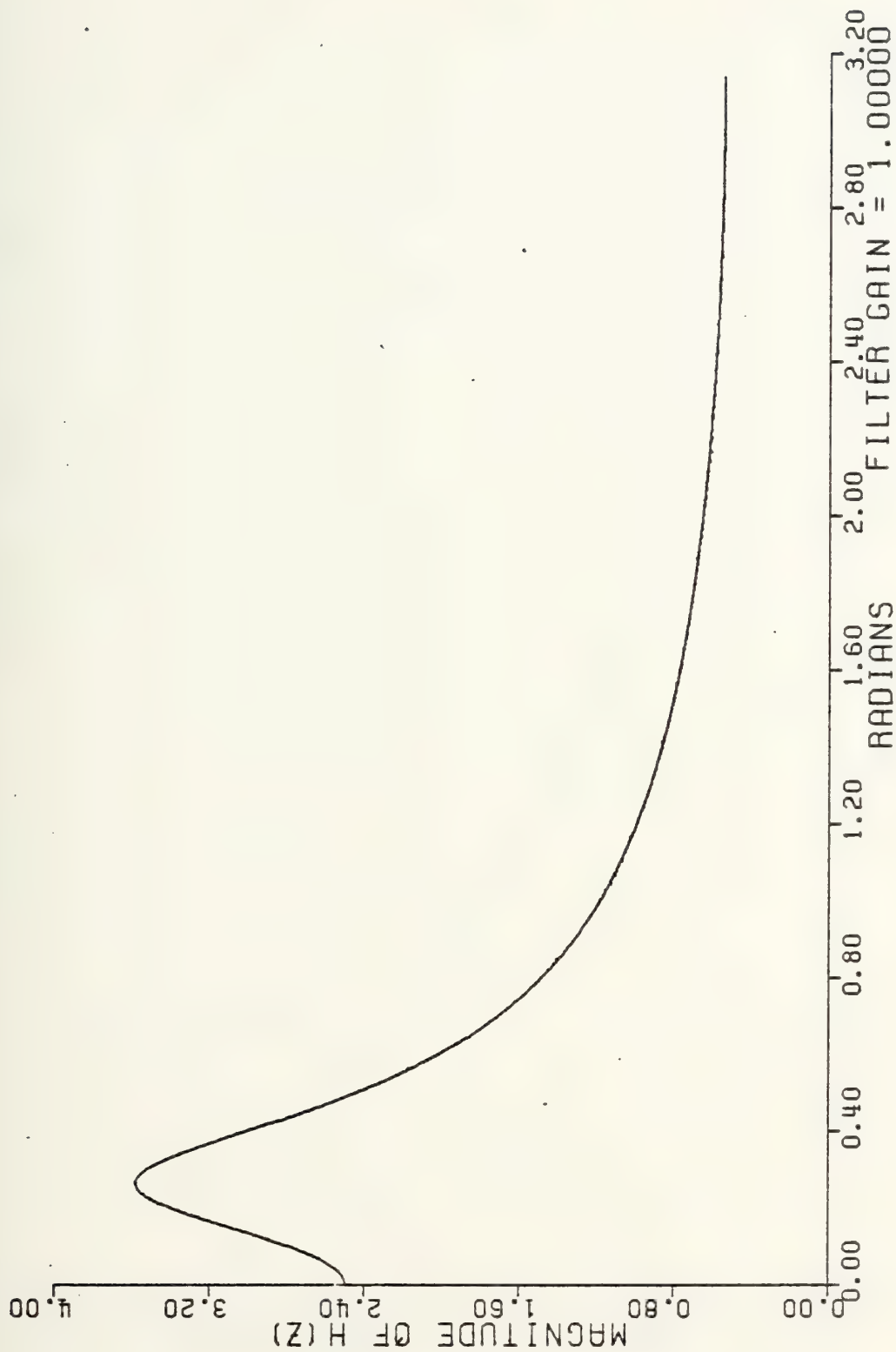


Figure 6-7f Magnitude Of  $H(z)$  For Example Two





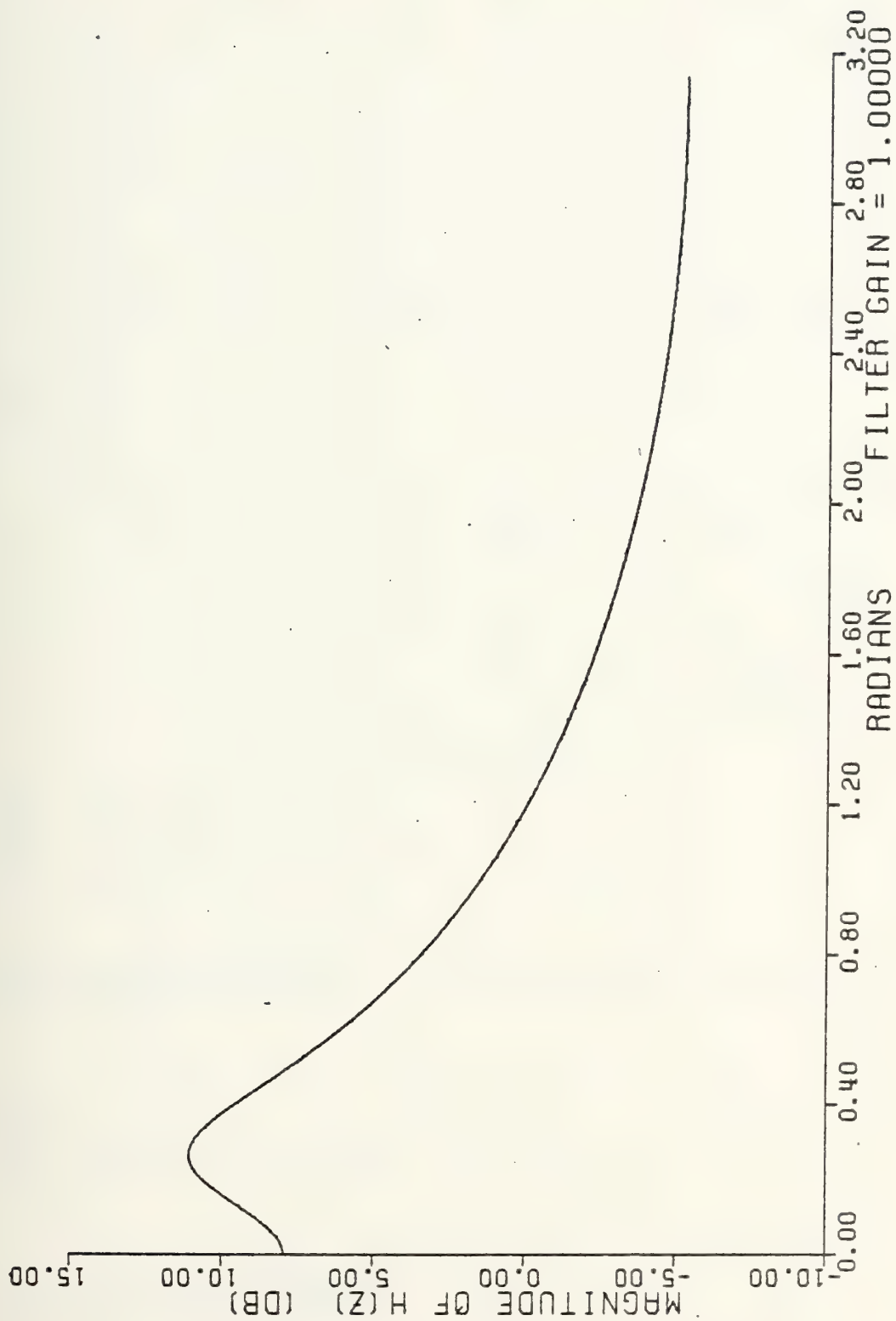


Figure 6-7g Magnitude Of The Transfer Function In Decibels For Example Two



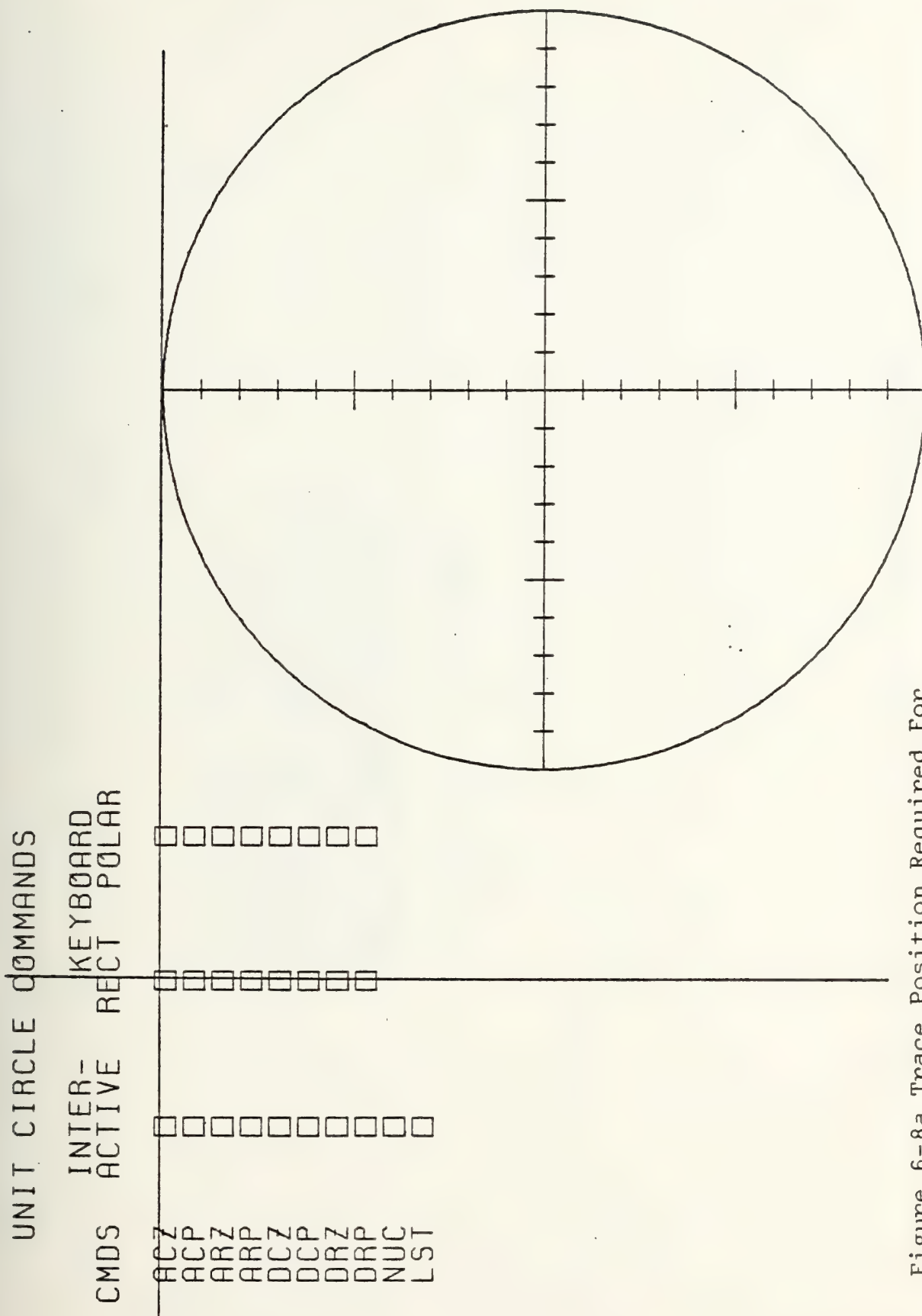


Figure 6-8a Trace Position Required For Executing The Command "Add A Complex Zero"-Rectangular Coordinates Mode



TO ADD A COMPLEX ZERO TO THE  
 DISPLAY, ENTER THE REAL AND  
 IMAGINARY PARTS WITHIN THE BOXES  
 PROVIDED. EACH NUMBER REQUIRES A  
 DECIMAL. REAL PART MINUS SIGNS  
 MUST BE INCLUDED WITHIN THE BOX.  
 NOTE: ALL COMPLEX ZEROS MUST BE  
 WITHIN TEN UNITS OF THE ORIGIN.

REAL  +/- J  IMAGINARY

Figure 6-8b Screen Display For The Command ACZRCT



TO ADD A COMPLEX ZERO TO THE  
 DISPLAY, ENTER THE REAL AND  
 IMAGINARY PARTS WITHIN THE BOXES  
 PROVIDED. EACH NUMBER REQUIRES A  
 DECIMAL. REAL PART MINUS SIGNS  
 MUST BE INCLUDED WITHIN THE BOX.  
 NOTE: ALL COMPLEX ZEROS MUST BE  
 WITHIN TEN UNITS OF THE ORIGIN.

|      |     |   |           |
|------|-----|---|-----------|
| REAL | +/- | J | IMAGINARY |
| -1.0 |     |   | 0.0       |

THE NEW COMPLEX ZERO WILL BE -1.000000 +/- J 0.0  
 IF CORRECT TYPE 'Y', IF NOT 'N'

Figure 6-8c Screen Display After Entering The First Complex Zero





ask whether or not the root location has been correctly entered. The user responds 'y' for yes, 'n' for no. If the root has been incorrectly entered, an 'n' response will rewrite the screen for another attempt. Upon receiving a 'y' response ASDF rewrites the unit circle and command array, and plots the new root. The second complex zero is entered exactly as the first.

The complex poles are entered as were the zeros. The trace cursor is first placed in the box ACPRCT. (Add a Complex Pole - Rectangular Coordinates), as in figure 6-9. Figure 6-10 shows the unit circle after all poles and zeros have been entered. The POLZRO command is terminated by aligning the trace cursor with any box, typing a character, waiting for the flashing alphanumeric cursor to enter the box, and typing "space", "x". The system responds by asking about the filter gain. In this example the filter gain is to be changed to .001836, so the correct response is 'y'. The new filter gain, 0.001836, is typed to terminate the POLZRO command.

To view the filter responses, the user enters the command:

#### RESPONSE

when the \*ASDF-RE signature appears. The hard copy responses of the filter are shown in figures 6-11a through 6-11g.



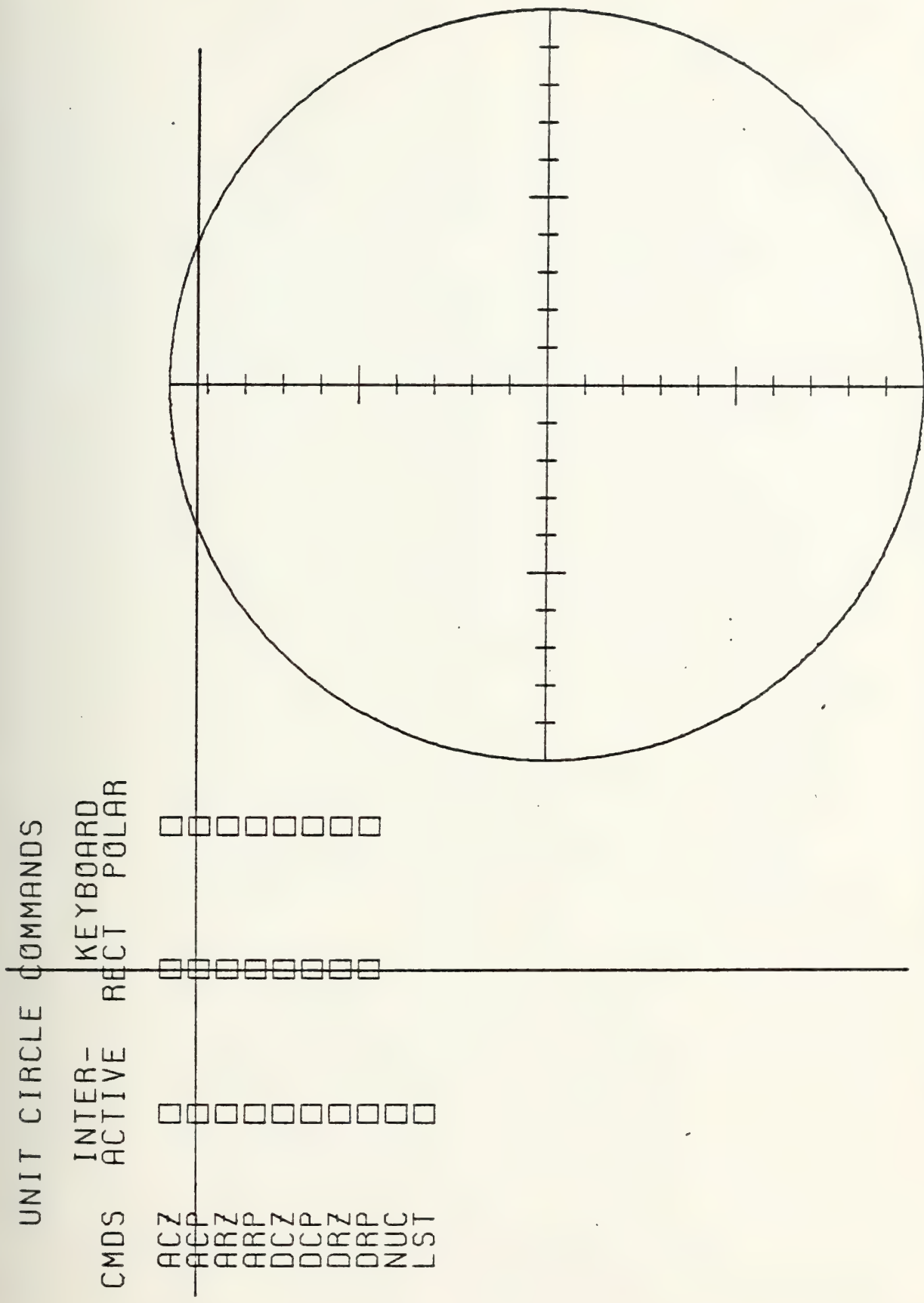


Figure 6-9 Trace Cursor Position To Execute Command ACPRCT



# UNIT CIRCLE COMMANDS

|      | INTER-<br>ACTIVE | KEYBOARD<br>RECT | POLAR |
|------|------------------|------------------|-------|
| CMDS |                  |                  |       |

|     |                          |                          |                          |
|-----|--------------------------|--------------------------|--------------------------|
| ACZ | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| ACP | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| ARZ | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| ARP | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DCZ | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DCP | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DRZ | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DRP | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| NUC | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| LST | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

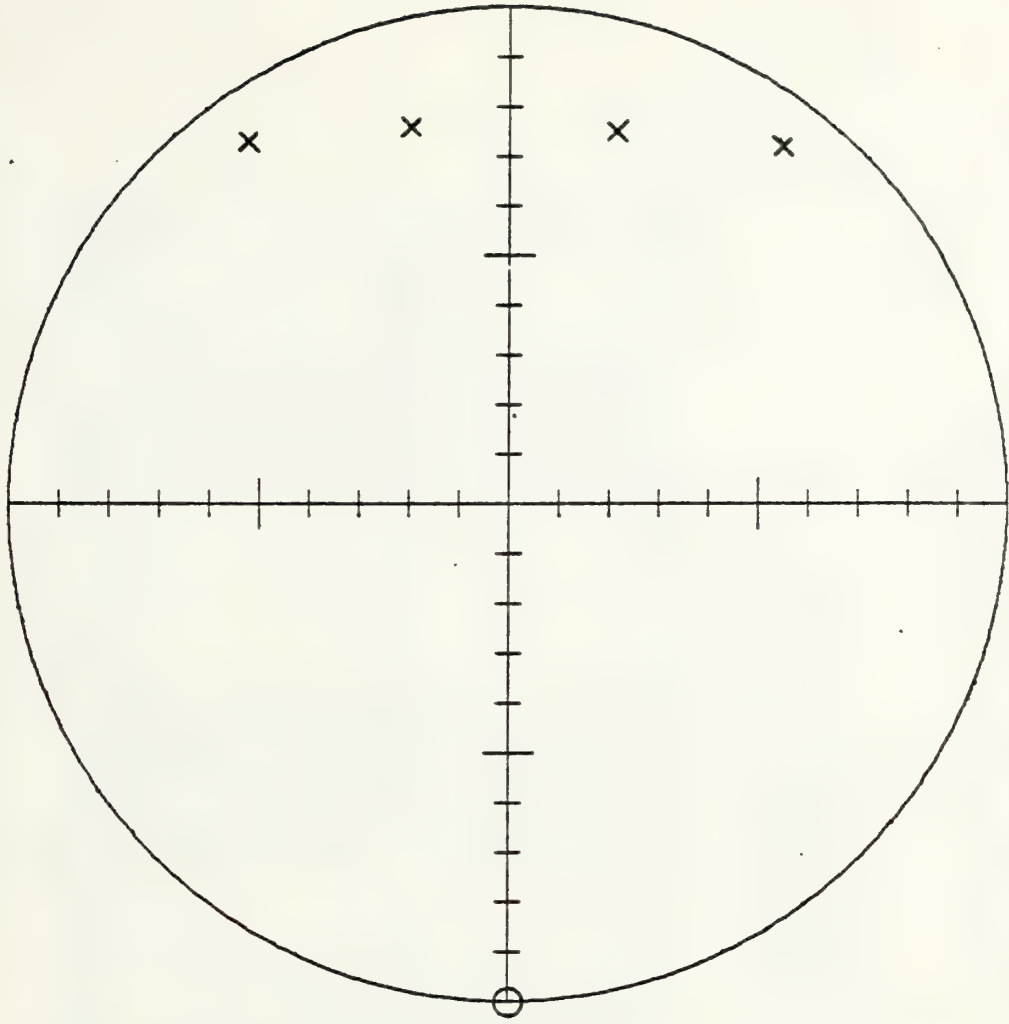


Figure 6-10 Unit Circle After All Poles and Zeros Have Been Returned-Example Two









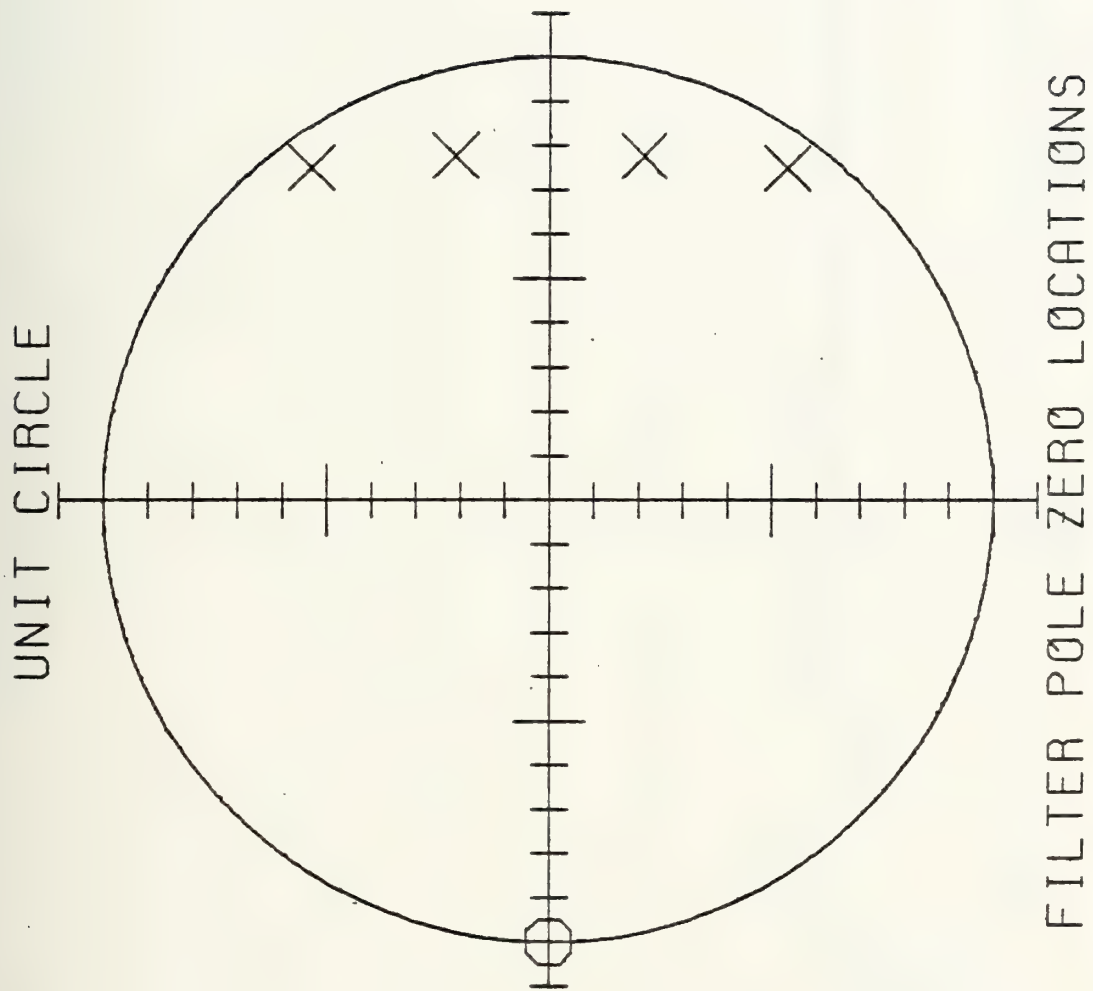


Figure 6-11b Pole Zero Locations For Example Two



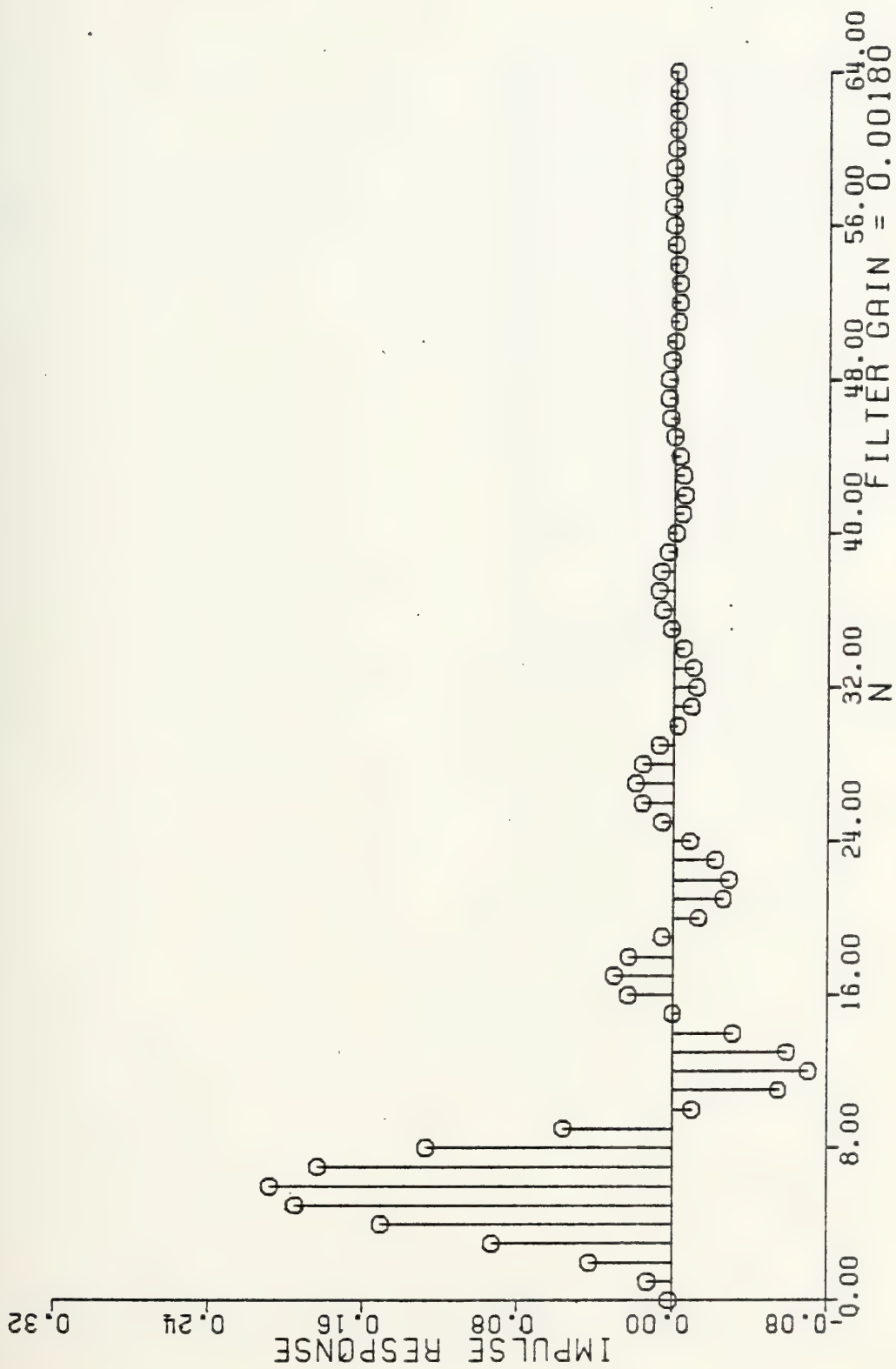


Figure 6-11c Unit Sample Response For Example Two



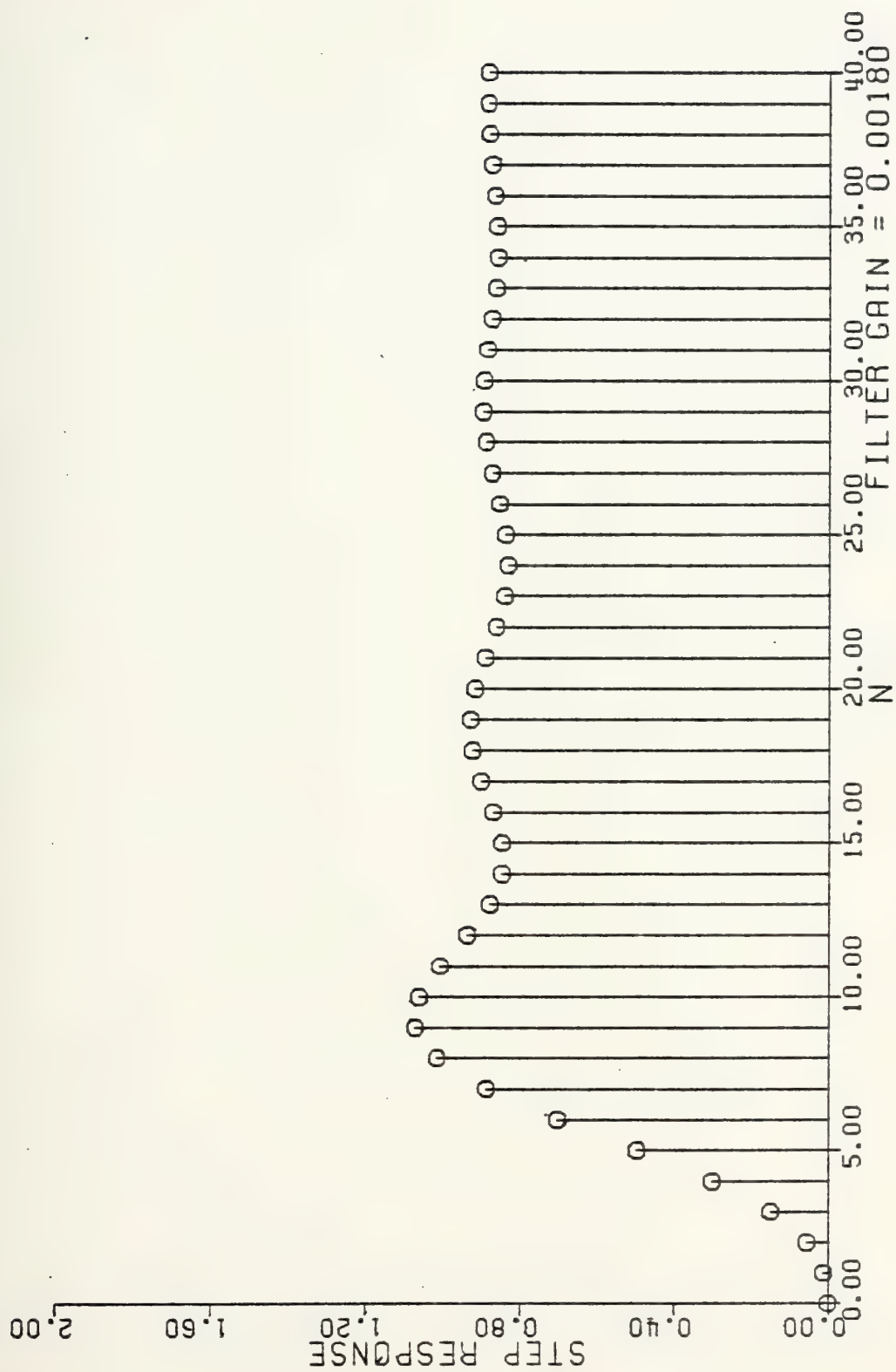


Figure 6-11d Unit Step Response For Example Two



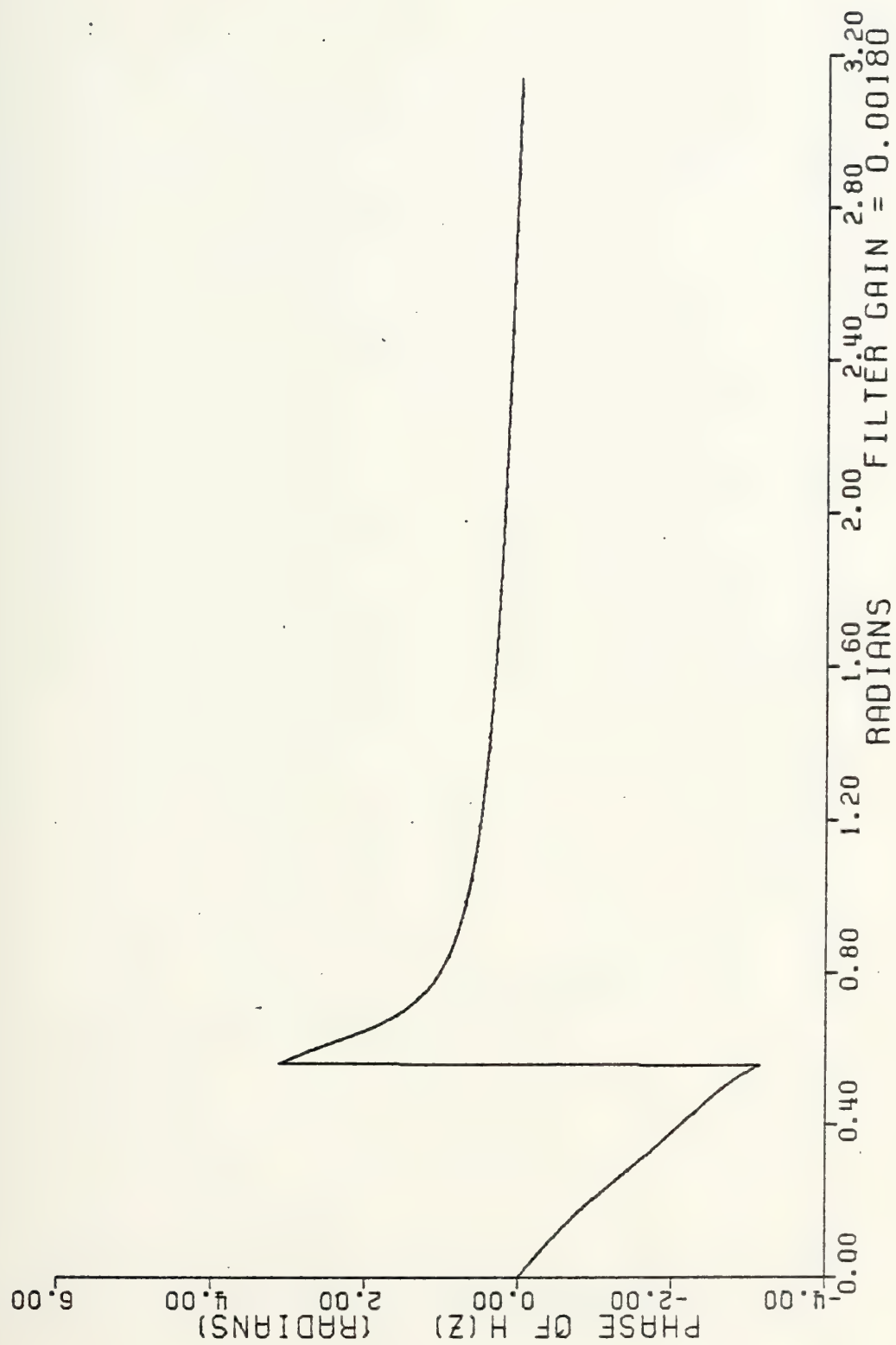


Figure 6-11e Phase Response For Example Two





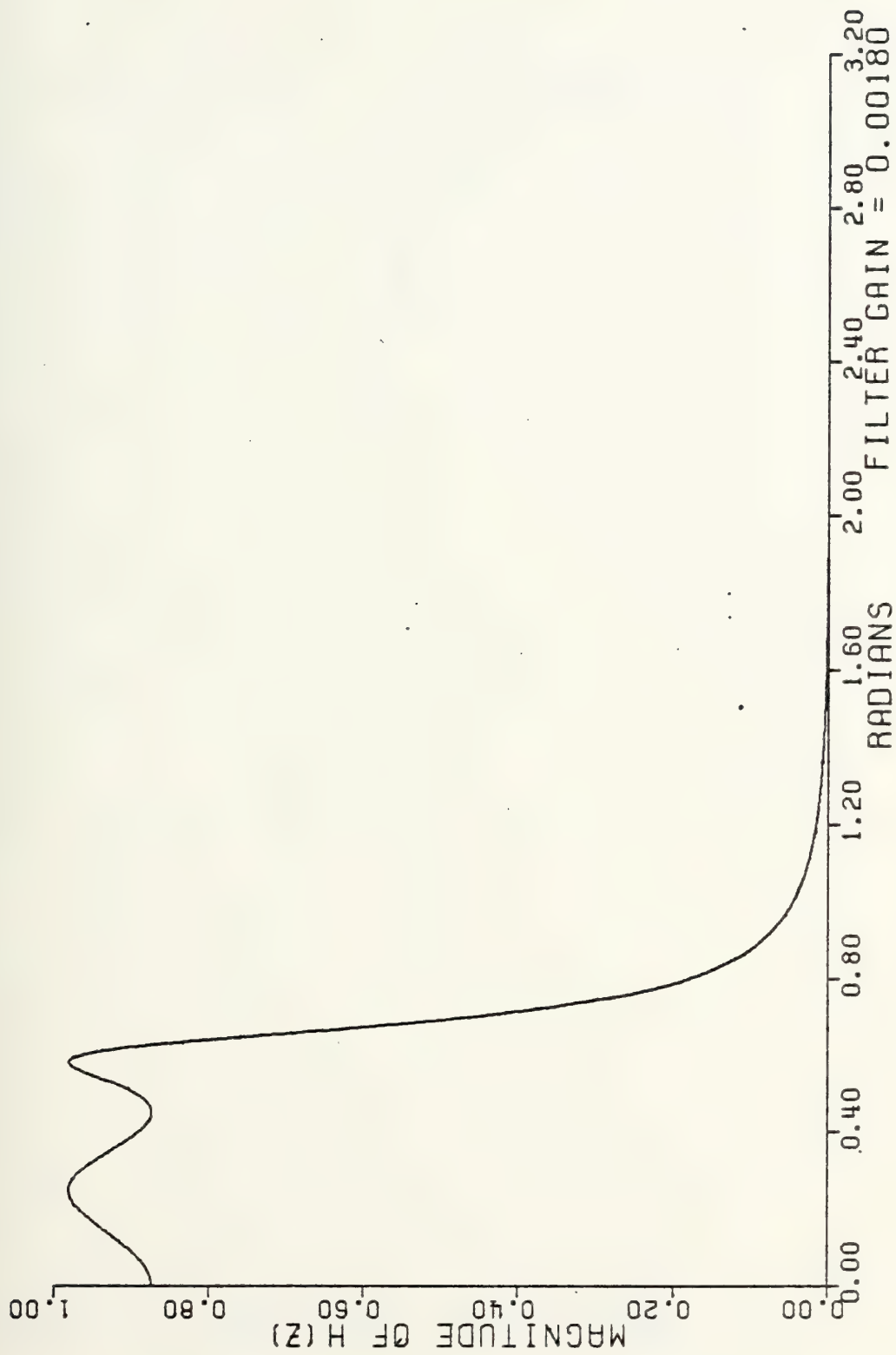


Figure 6-11f Magnitude of  $H(z)$  For Example Two



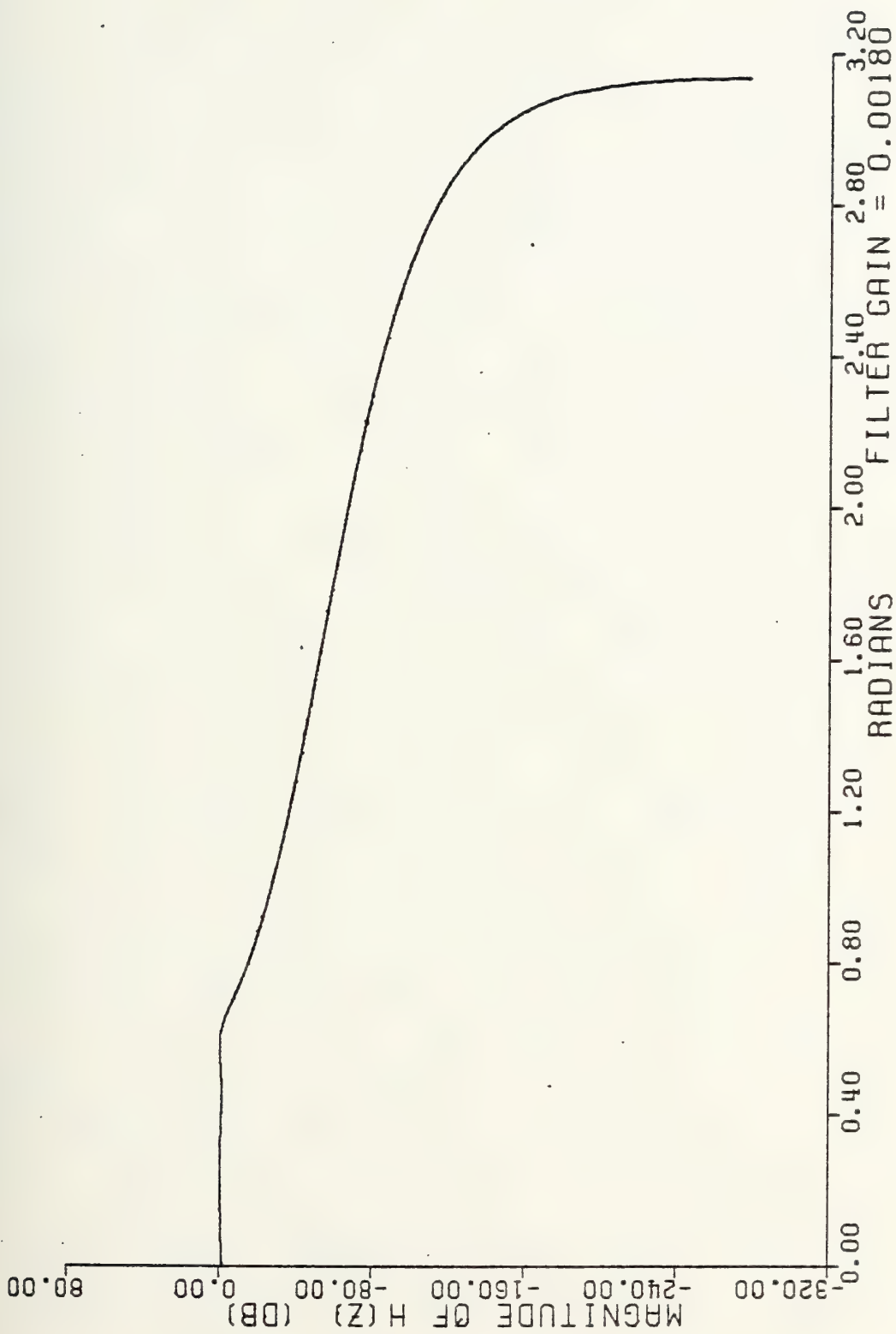


Figure 6-11g Magnitude Of  $H(z)$  In Decibels For Example Two



### C. EXAMPLE THREE - POLAR MODE

This example uses the polar mode to generate an all pass filter. The filter complex pole pairs are to be located at (0.2 at +/- 45 degrees), and (0.5 at +/- 30 degrees). The filter zeros are to be at (5.0 at +/- 45 degrees), and (2.0 at +/- 30 degrees). The filter gain is to be 0.01. The first step is to change the specifications in degrees to radians. Thirty degrees equals .5235 radians. Forty five degrees is .7854 radians. The entry of poles and zeros exactly parallels the above examples, however, the desired commands are ACPPLR, (Add a Complex Pole - Polar Coordinates), and ACZPLR, (Add a Complex Zero - Polar Coordinates). Figure 6-12a shows the trace cursor position for ACPPLR. Figure 6-12b shows the trace cursor position for ACZPLR. The angular part of any pole or zero entered in the polar mode must be in radians. Figure 6-12c shows the entry of the first complex zero pair. Figure 6-12d shows the entry of the first complex pole pair. Figure 6-12e shows the change in filter gain from 1.00 to 0.01. Figures 6-13a through 6-13g show the hard copy responses for this filter. It should be noted that zeros which plot outside of the unit circle are indicated by stars. All such zeros are plotted at a radius of 1.1 along the same angular component as the actual zero location. Zeros are plotted this way for two reasons: first to make the output display consistent for all relevant filters, and second to emphasize zeros which



| UNIT CIRCLE COMMANDS |                          |                          |                          |
|----------------------|--------------------------|--------------------------|--------------------------|
| CMDS                 | INTER-<br>ACTIVE         | KEYBOARD<br>RECT         | POLAR                    |
| ACZ                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| ACP                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| ARZ                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| ARP                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DCZ                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DCP                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DRZ                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| DRP                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| NUC                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| LST                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

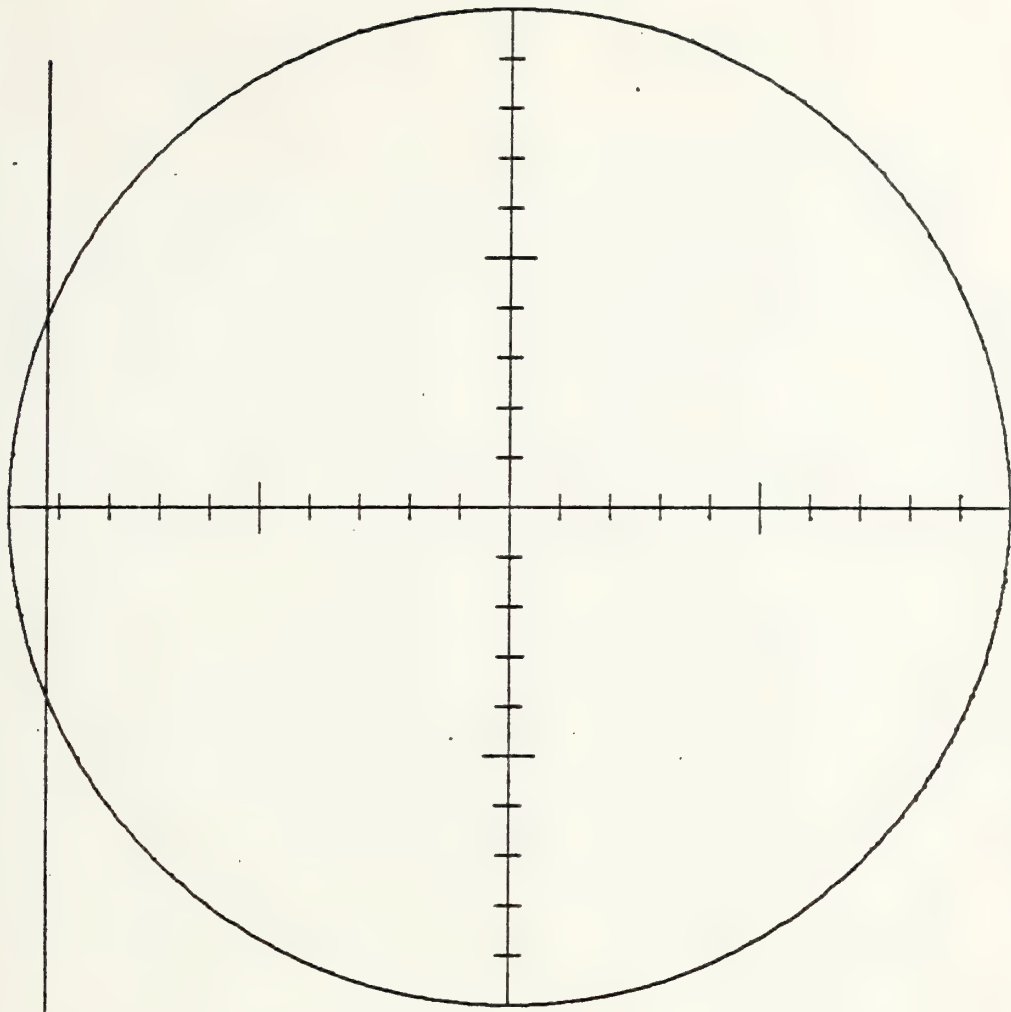


Figure 6-12a Trace Cursor Position For The ACPPLR Command





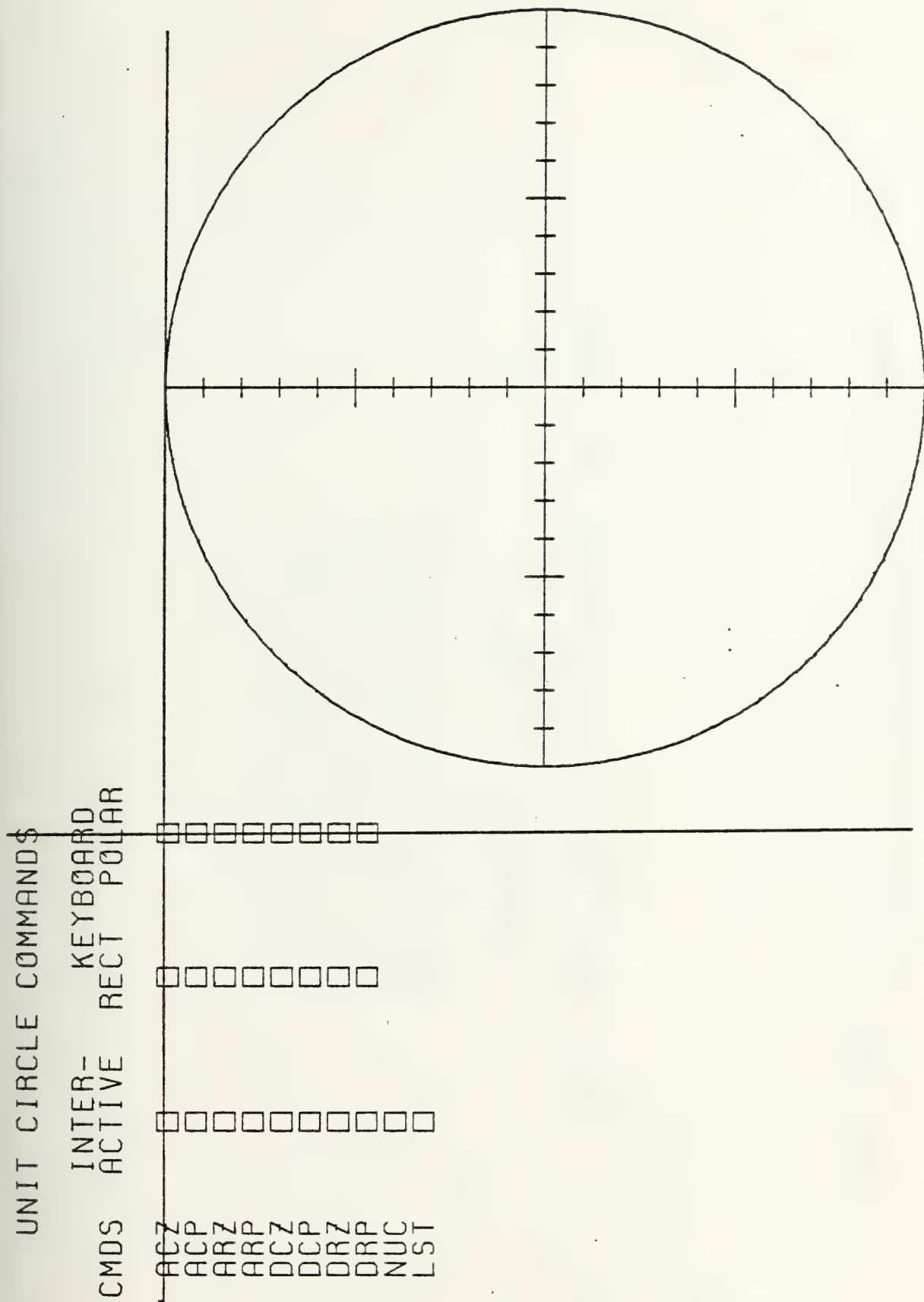


Figure 6-12b Trace Cursor Position For The ACZPLR Command



TO ADD A COMPLEX ZERO TO THE  
 DISPLAY, ENTER THE RADIAL AND  
 THETA COMPONENTS WITHIN PROVIDED  
 BOXES. THE MAGNITUDE OF RHO MUST  
 BE LESS THAN TEN, THETA(RADIANS)  
 EACH NUMBER REQUIRES A DECIMAL,  
 AND AS APPROPRIATE A MINUS SIGN.  
 BOTH ARE CONFINED TO THE BOXES.

| MAGNITUDE | ANGLE                  |
|-----------|------------------------|
| >> 15.0   | +/- <del>4</del> .7854 |

THE NEW COMPLEX ZEROS WILL BE 15.0000000 +/- ~~4~~ 0.7854000  
 IF CORRECT TYPE 'Y', IF NOT 'N'.

Figure 6-12c Screen Display After Entering The First Complex  
 Zero Pair In Polar Coordinates



TO ADD A COMPLEX POLE WITHIN THE UNIT CIRCLE ENTER THE RADIAL AND THETA COMPONENTS WITHIN PROVIDED BOXES. RHO MUST BE ONE OR LESS. THETA MUST BE ENTERED IN RADIAN'S EACH NUMBER REQUIRES A DECIMAL, AND AS APPROPRIATE A MINUS SIGN. BOTH ARE CONFINED TO THE BOXES.

MAGNITUDE      ANGLE  
 >>  +/- ~~X~~

THE NEW COMPLEX POLES WILL BE 0.20000000 +/- ~~X~~ 0.78540000  
 IF CORRECT TYPE 'Y', IF NOT 'N'.

Figure 6-12d Screen Display After Entering The First Complex Pole Pair In Polar Coordinates



THE FILTER GAIN IS CURRENTLY 1.00000000  
DO YOU WISH TO CHANGE THE FILTER GAIN?  
(TYPE Y OR N)  
TYPE IN THE NEW FILTER GAIN 12 OR FEWER NUMBERS  
INCLUDE A DECIMAL POINT  
>0.01  
NEW FILTER GAIN IS: 0.01000000

Figure 6-12e Screen Representation After Changing Filter Gain









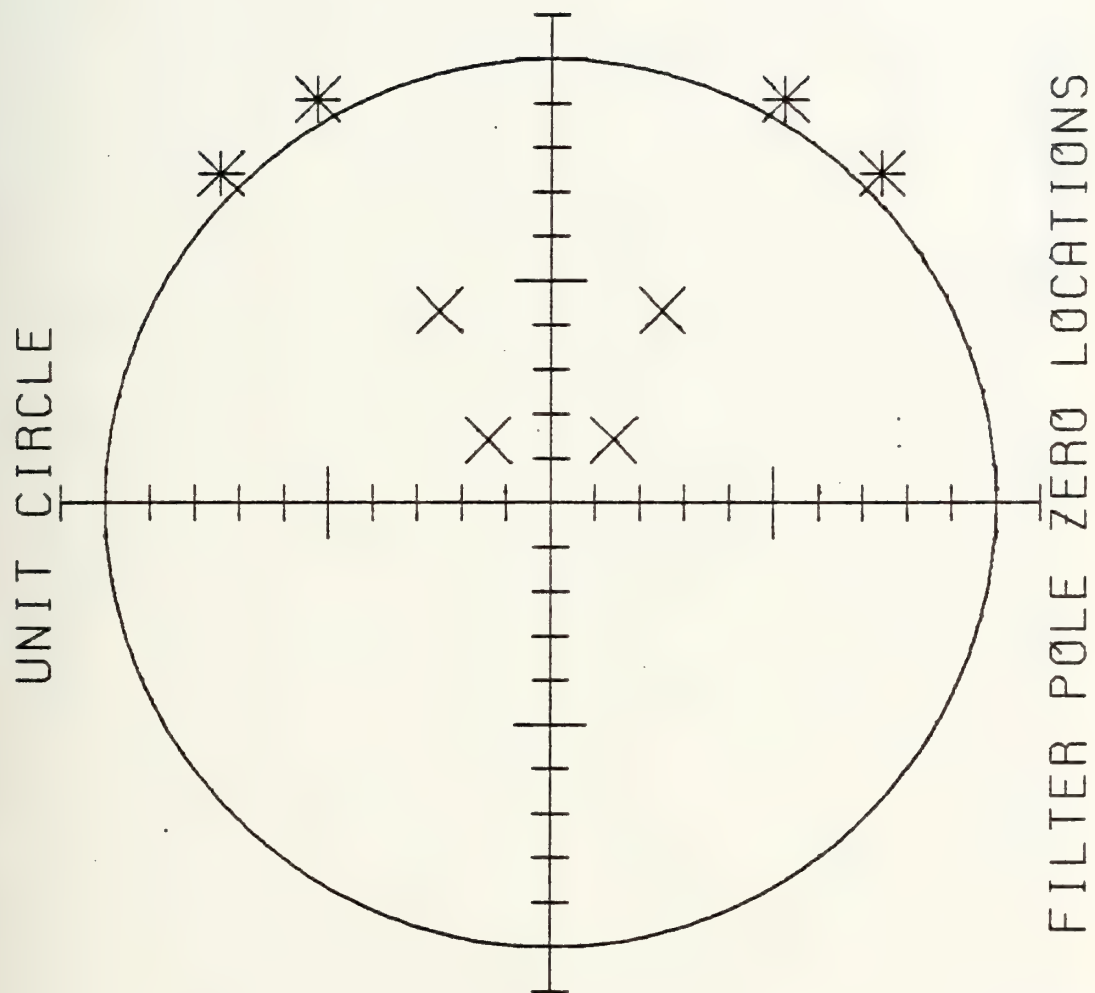


Figure 6-13b Pole Zero Location For Example Three



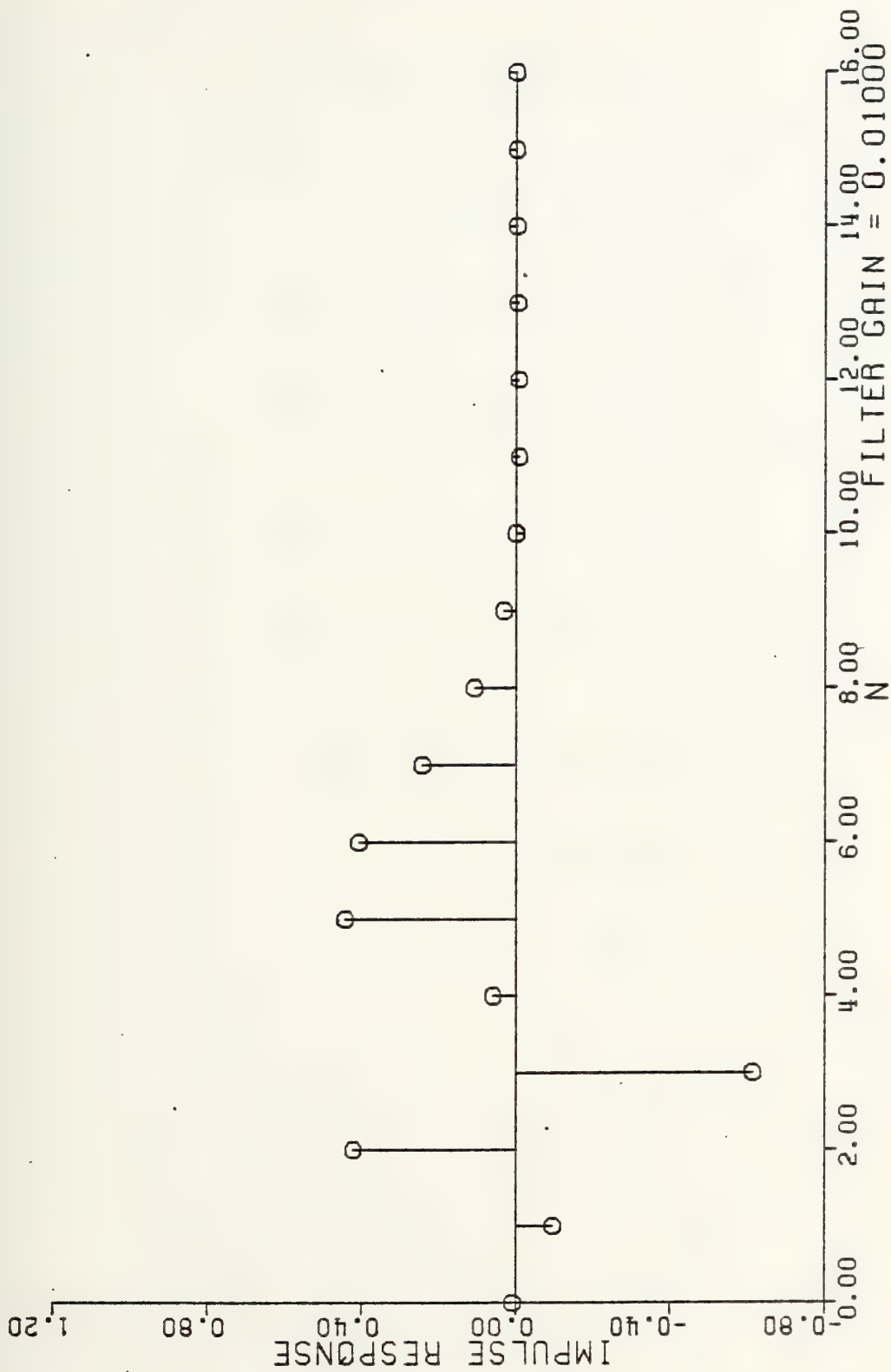


Figure 6-13c Unit Sample Response For Example Three



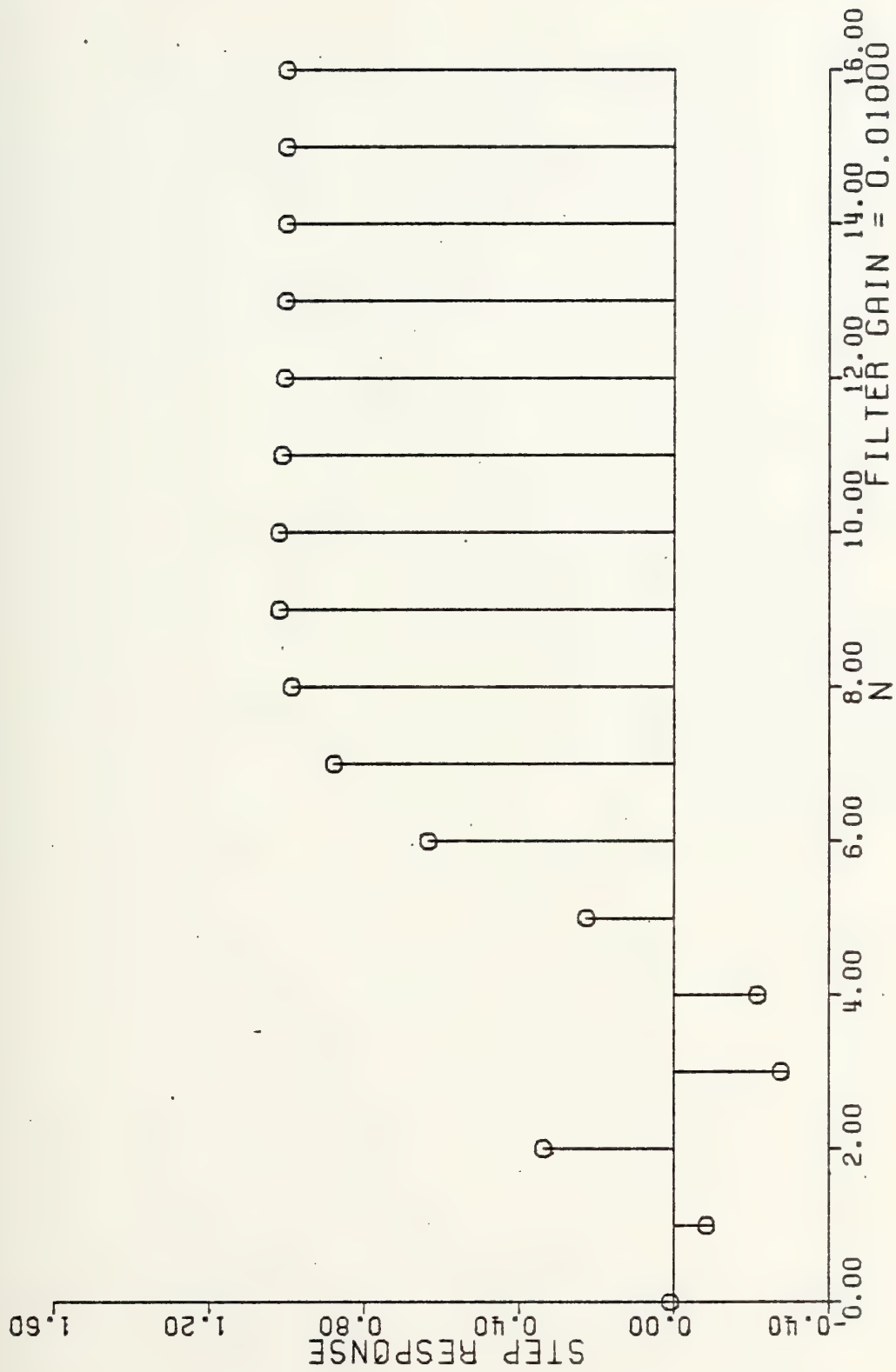


Figure 6-13d Unit Step Response For Example Three





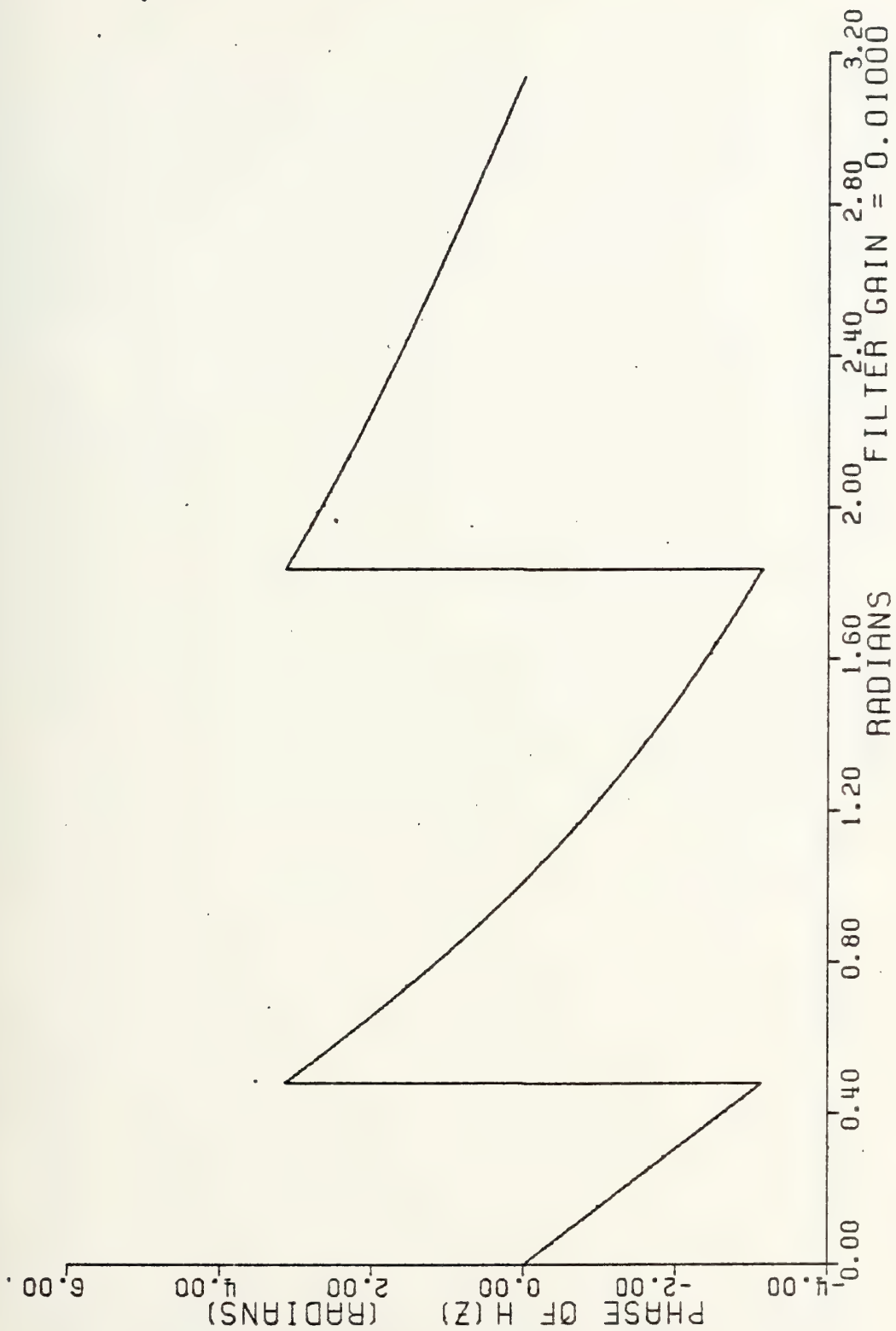


Figure 6-13e Phase Response For Example Three



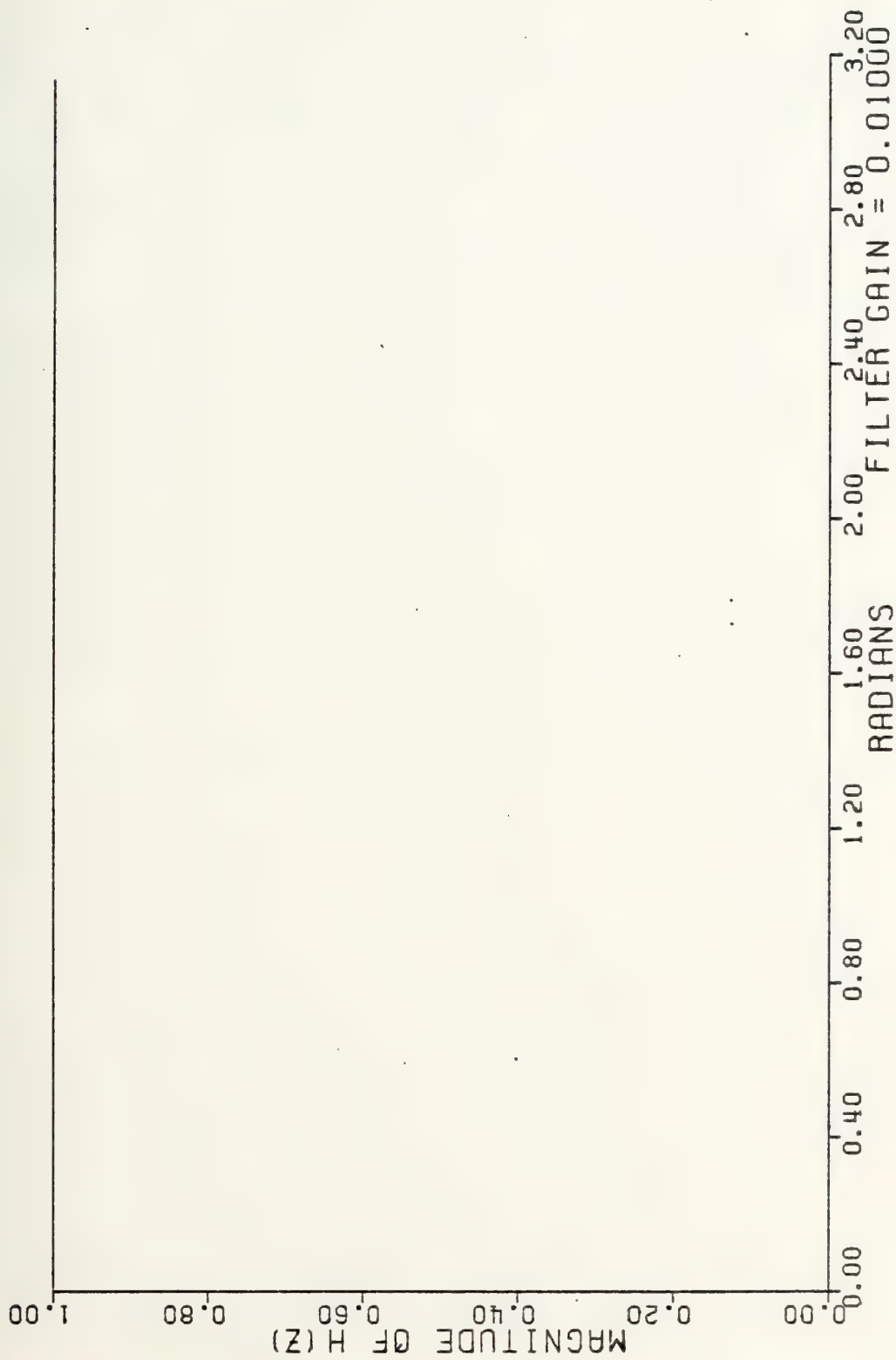


Figure 6-13f Magnitude of  $H(z)$  For Example Three



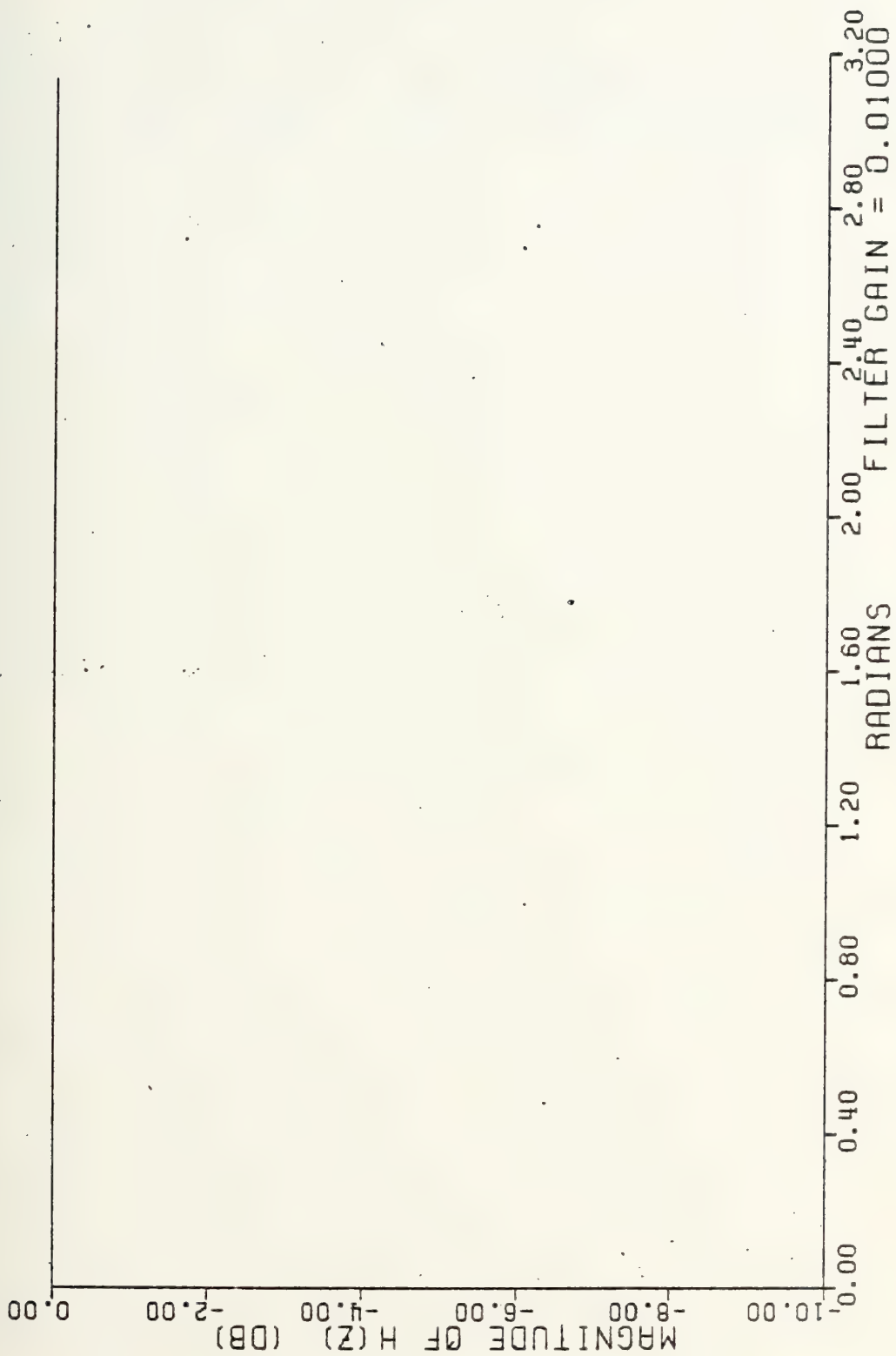


Figure 6-13g Magnitude Of  $H(z)$  In Decibels For Example Three



are close to, but outside, the unit circle. The importance of distinguishing between zeros very near the unit circle, either interior or exterior, rests on the extreme sensitivity of the phase of the frequency response as a zero moves from the interior, onto, and to the exterior of the unit circle.





## VII. ADDITIONAL EXAMPLES

The purpose of this chapter is threefold. The primary objective is to provide the user with a series of plots demonstrating the general form of those types of filters not considered in chapter six. Additionally, this chapter demonstrates the responses generated by poles located at a series of positions within the unit circle. It further provides examples for user comparison when learning to interact with the  $z$ -plane.

The filters considered in the examples of chapter six include a fourth-order low-pass filter and a second-order all-pass filter. The sequence of figures 7-1a through 7-1g shows the general pole zero placements for a high-pass filter, and associated responses. Similarly, figure 7-2a through 7-2g show the responses for a band-pass filter. Figures 7-3a through 7-3g show a band-reject or notch filter.

The secondary purpose of the chapter is to show the user the general forms of time and frequency responses as the filter poles move in the  $z$ -plane. The sequence of figures 7-4 through 7-11 shows the responses for a real pole at various locations on the real axis. Figures 7-12 through 7-19 similarly show the responses of a complex pole pair moving across the  $z$ -plane. Frequency responses are not



shown for poles on, or outside of, the unit circle since the frequency response is not defined for these cases. Figures 7-17, 7-20, and 7-21 show the change in responses for a pair of complex poles which move away from the real axis.

The final intent of the chapter is to provide the user with additional figures for comparison should he need further practice in manipulating poles and zeros in the  $z$ -plane. It is important to note that the ASDP program will not allow the user to position poles on or outside the unit circle. This limitation results in several plots which can not be reproduced by the user.







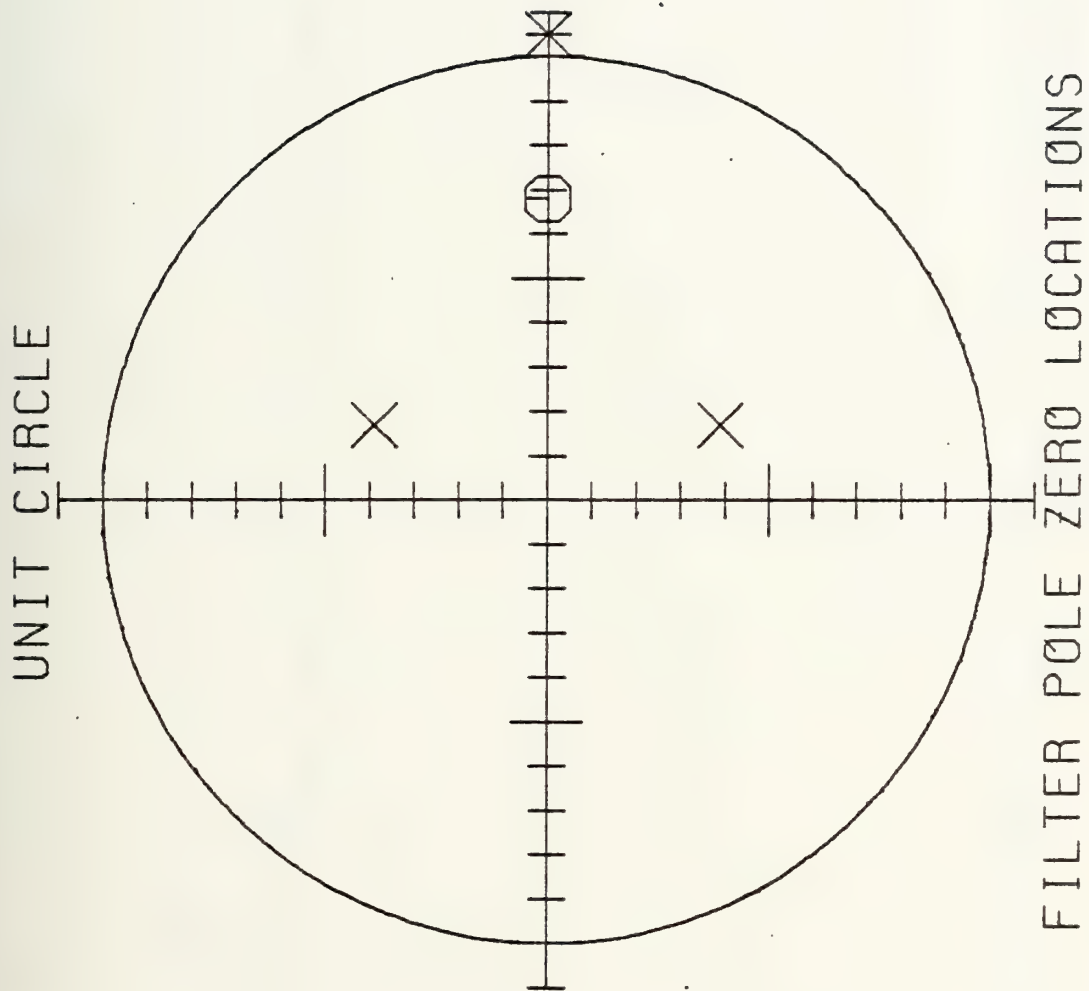


Figure 7-1b Plot Of Pole Zero Locations For A Second-Order High-Pass Filter





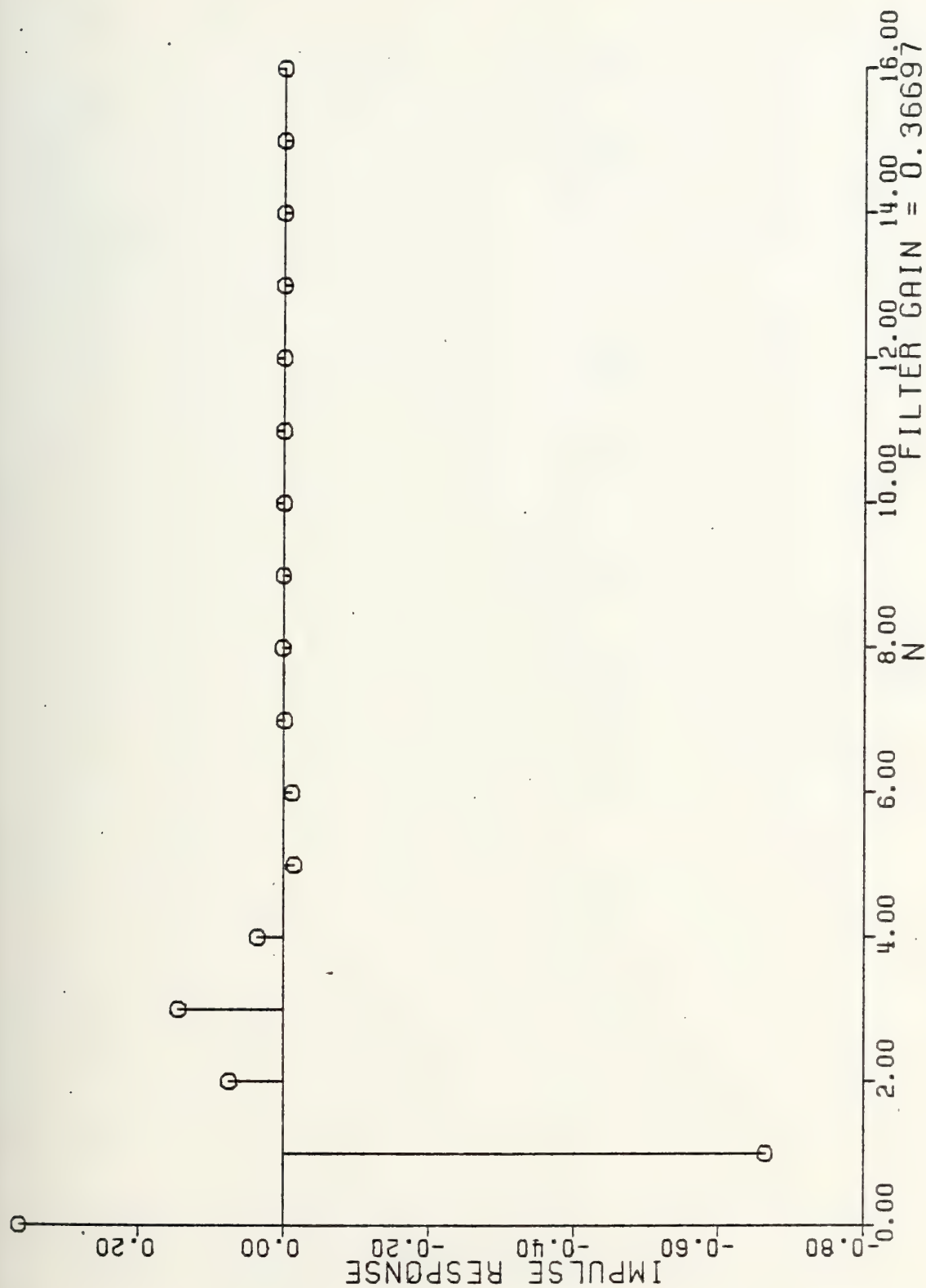


Figure 7-1c Unit Sample Response For A Second-Order High-Pass Filter



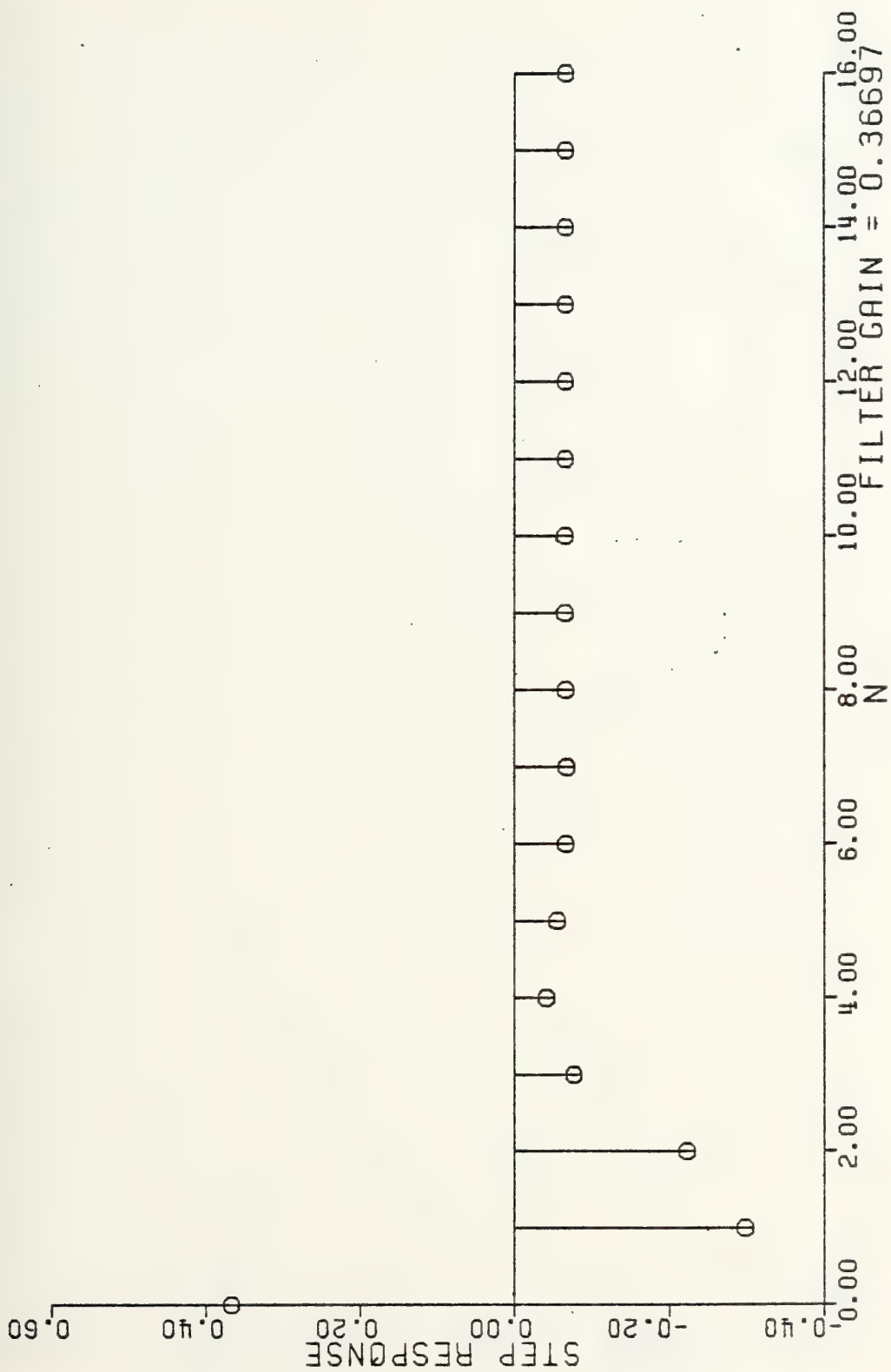


Figure 7-1d Unit Step Response For A Second-Order High-Pass Filter



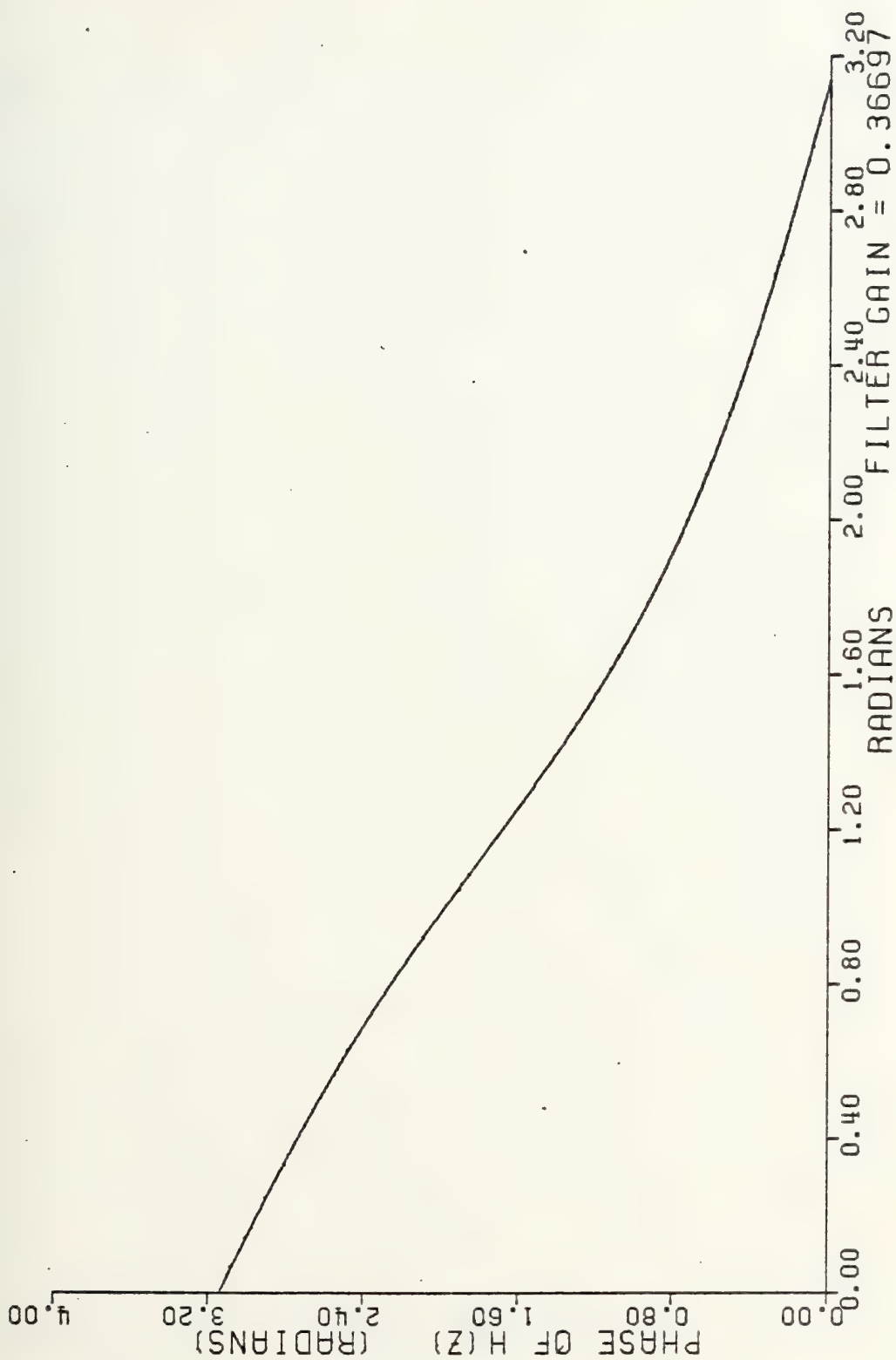


Figure 7-1e Phase Response For A Second-Order High-Pass Filter



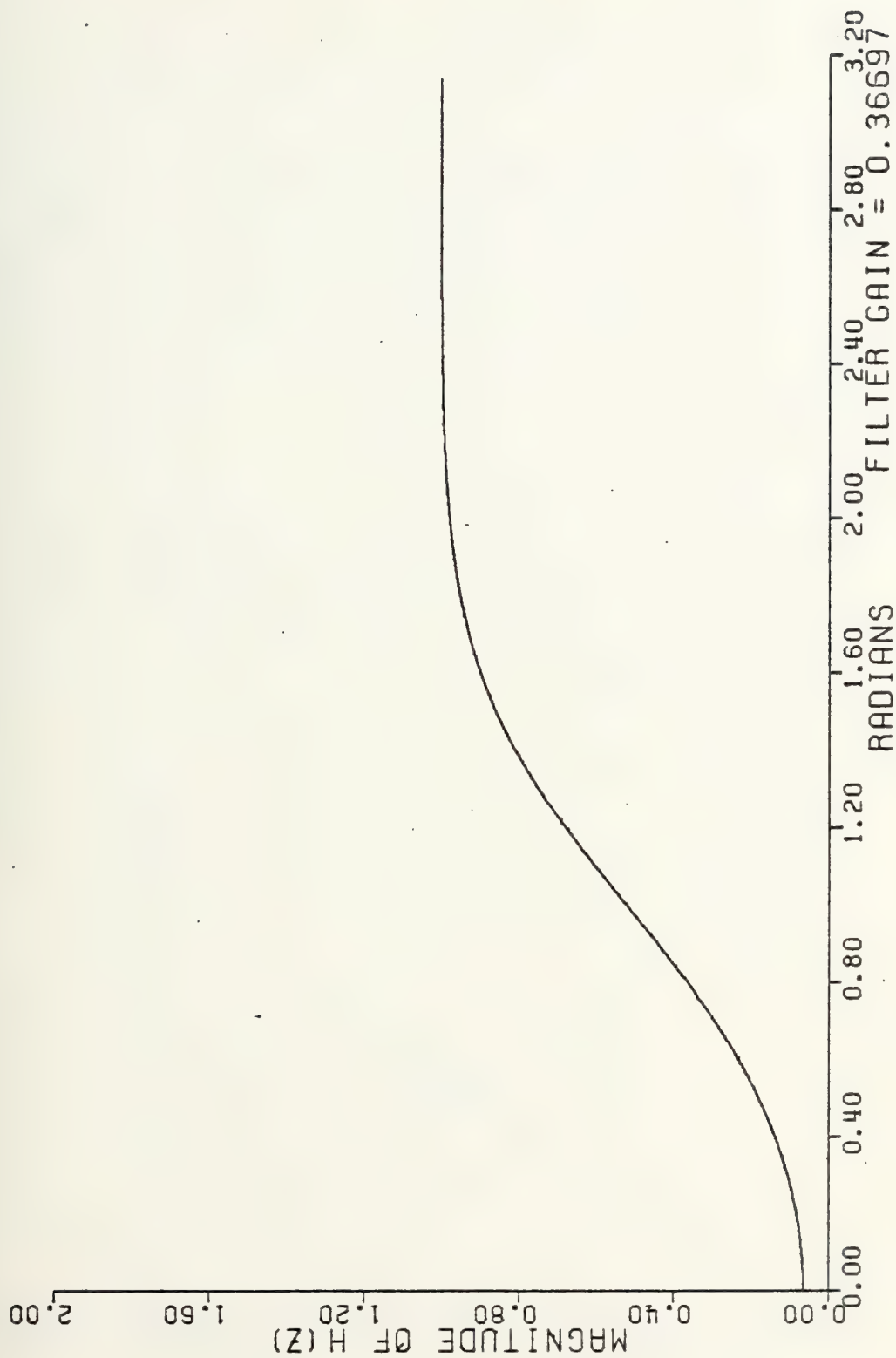


Figure 7-1f Magnitude of  $H(z)$  For A Second-Order High-Pass Filter





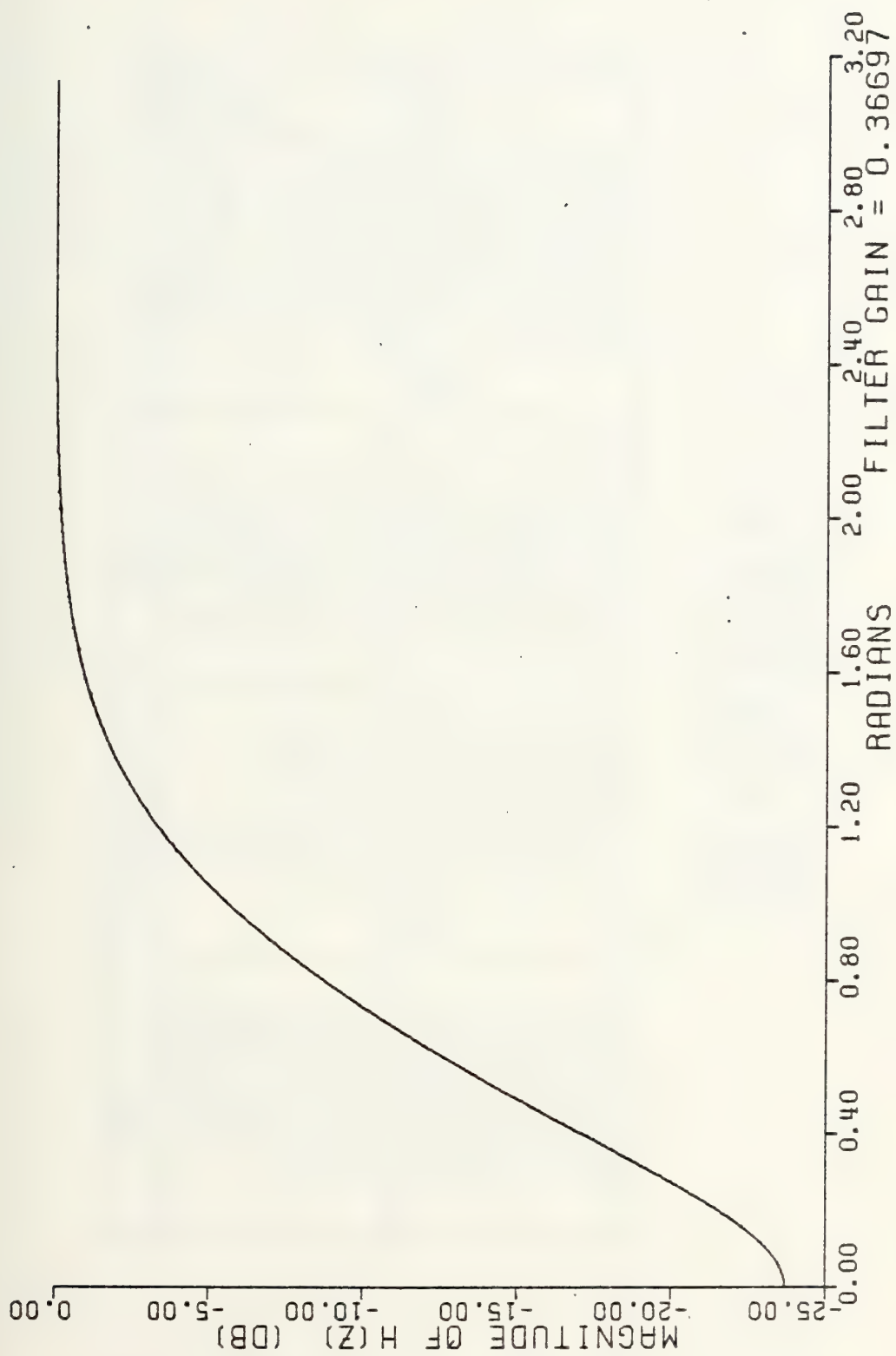


Figure 7-1g Magnitude Of  $H(z)$  In Decibels For A Second-Order High-Pass Filter







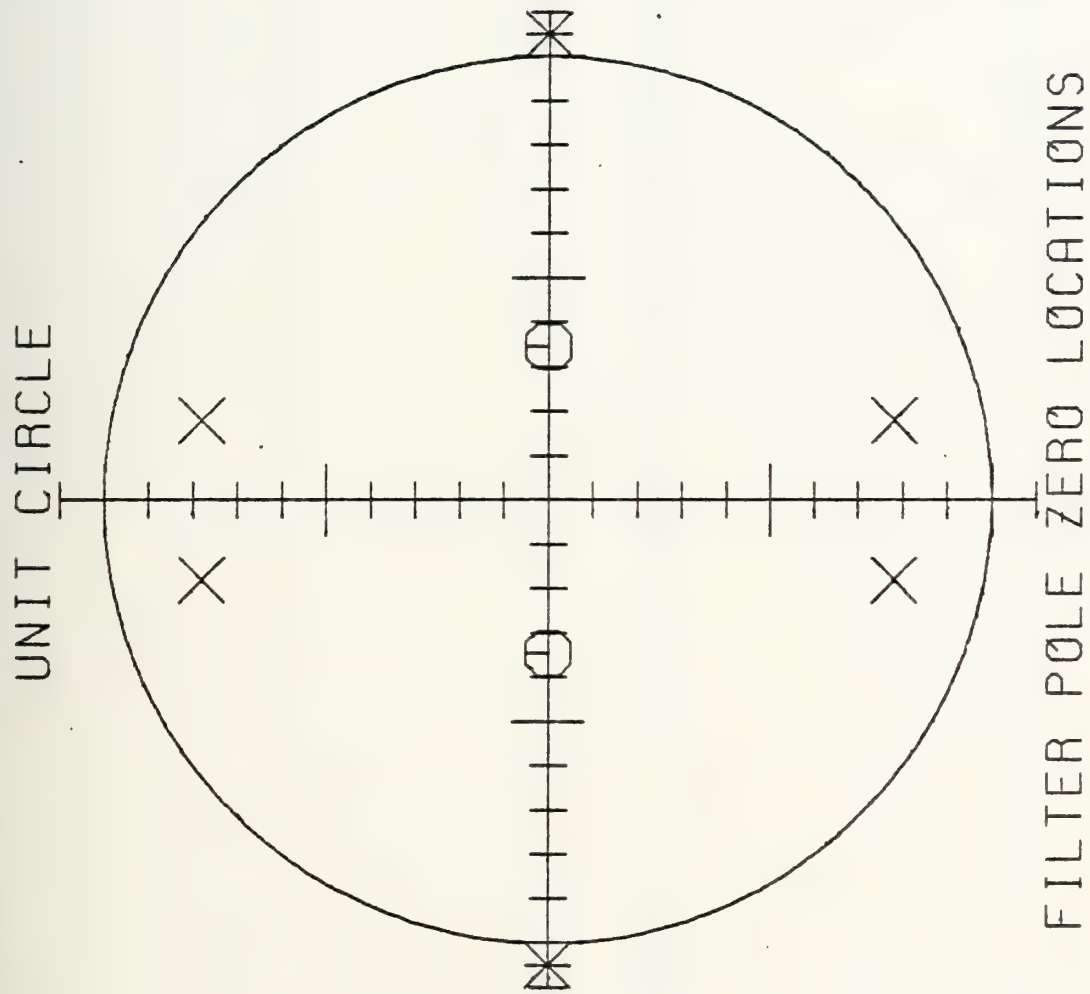


Figure 7-2b Plot Of Pole Zero Locations For A Fourth-Order Band-Pass Filter



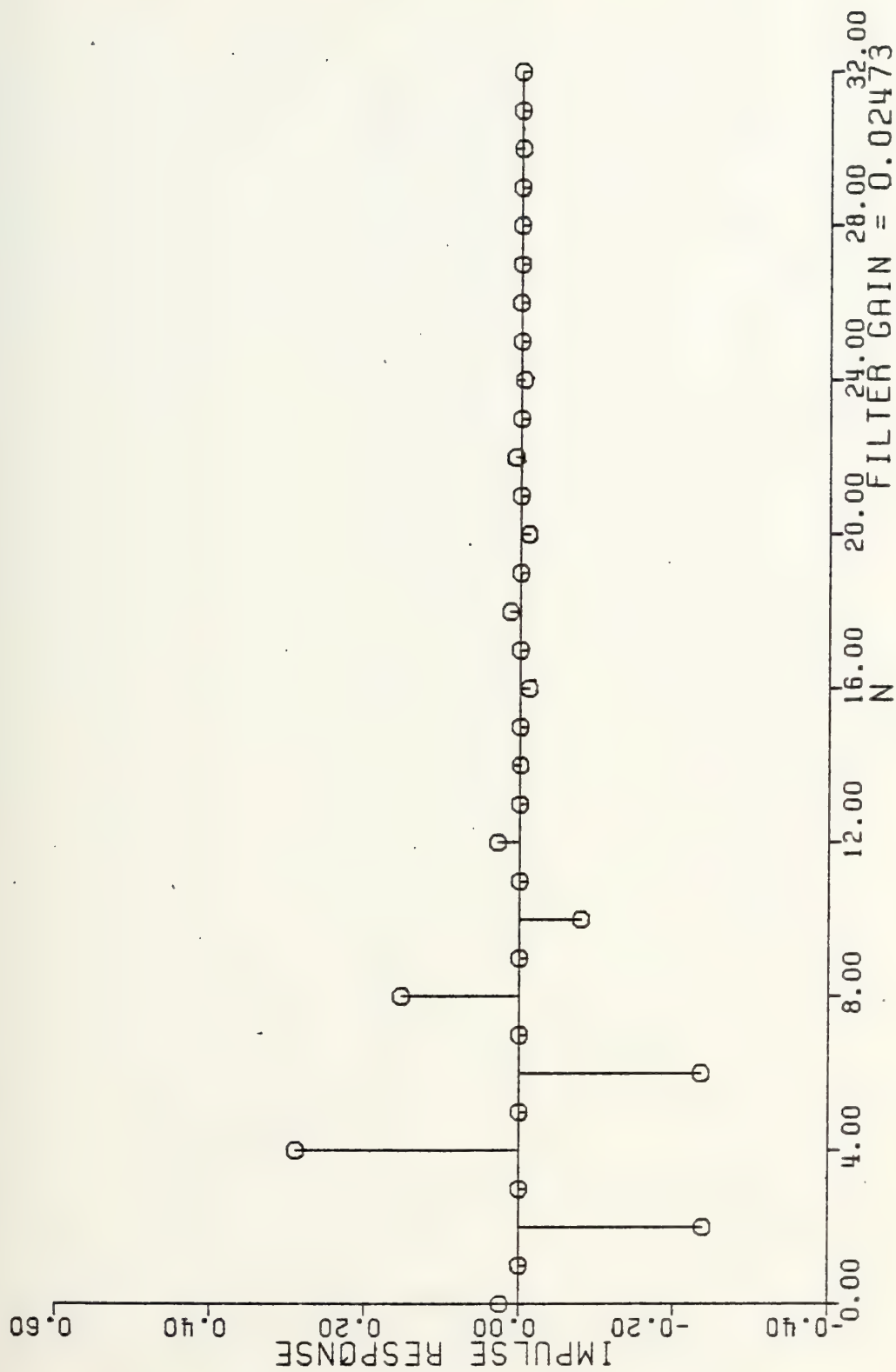


Figure 7-2c Unit Sample Response For A Fourth-Order Band-Pass Filter





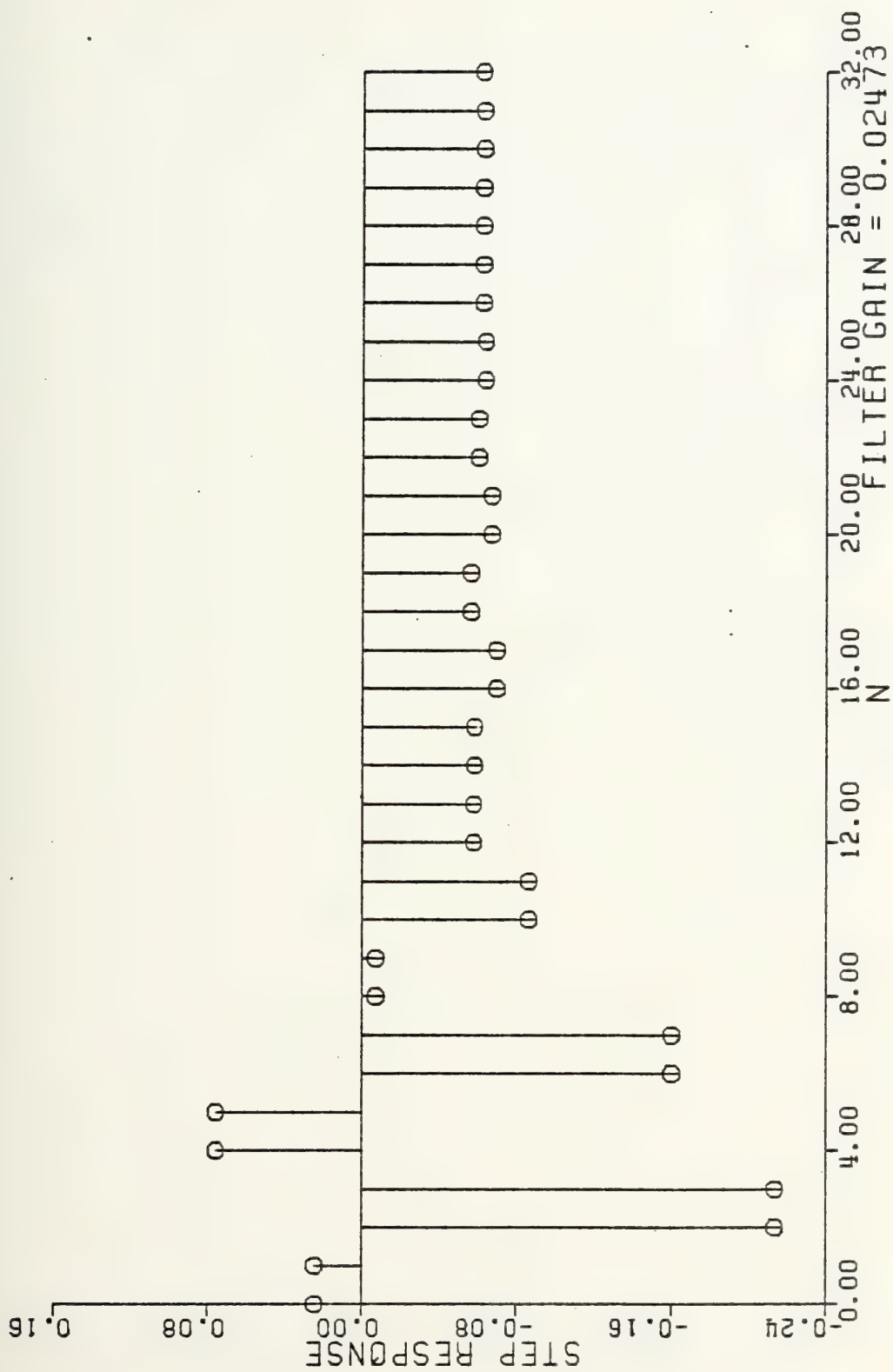


Figure 7-2d Unit Step Response For A Fourth-Order Band-Pass Filter



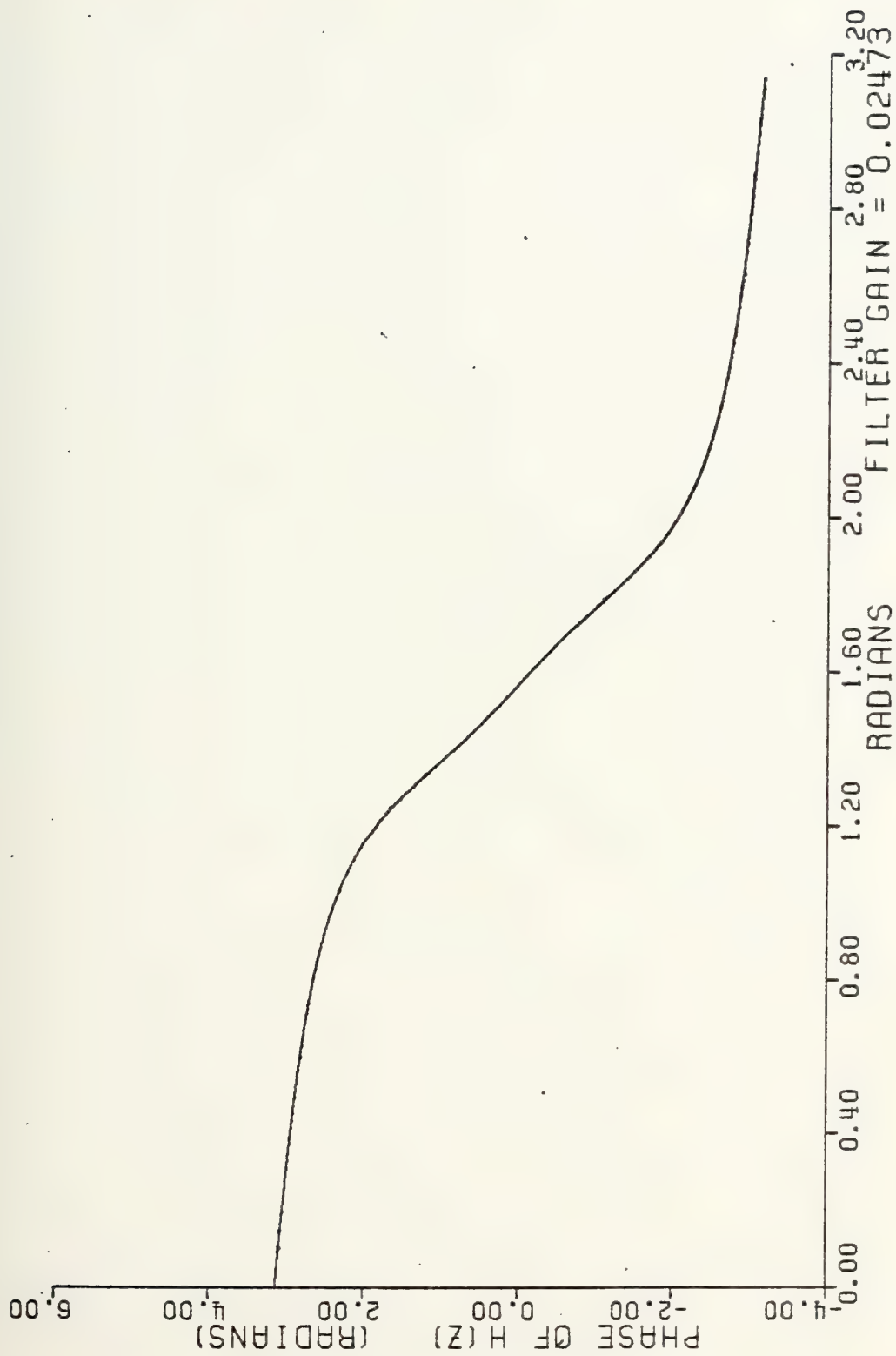


Figure 7-2e Phase Response For A Fourth-Order Band-Pass Filter



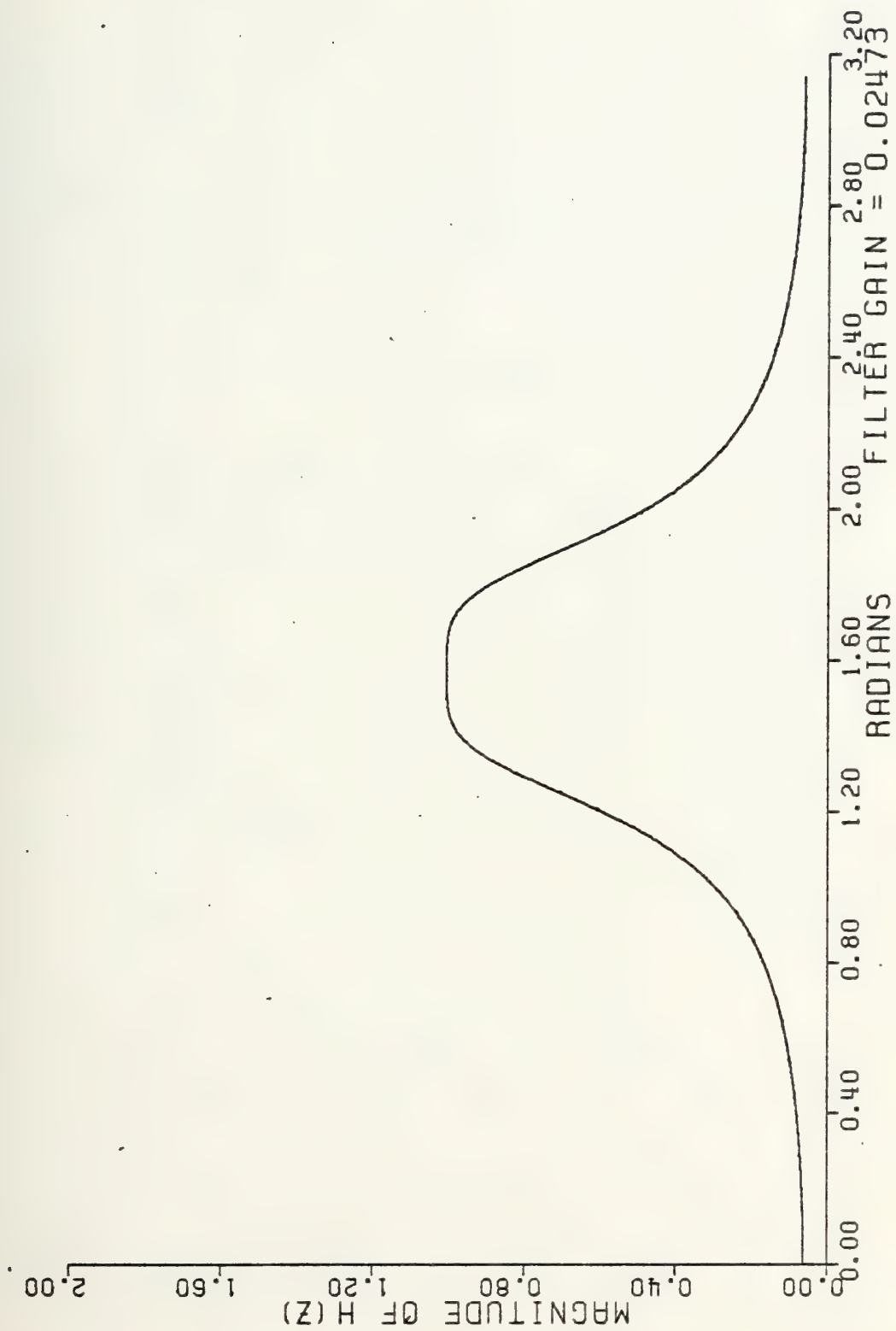


Figure 7-2f Magnitude Of  $H(z)$  For a Fourth-Order Band-Pass Filter



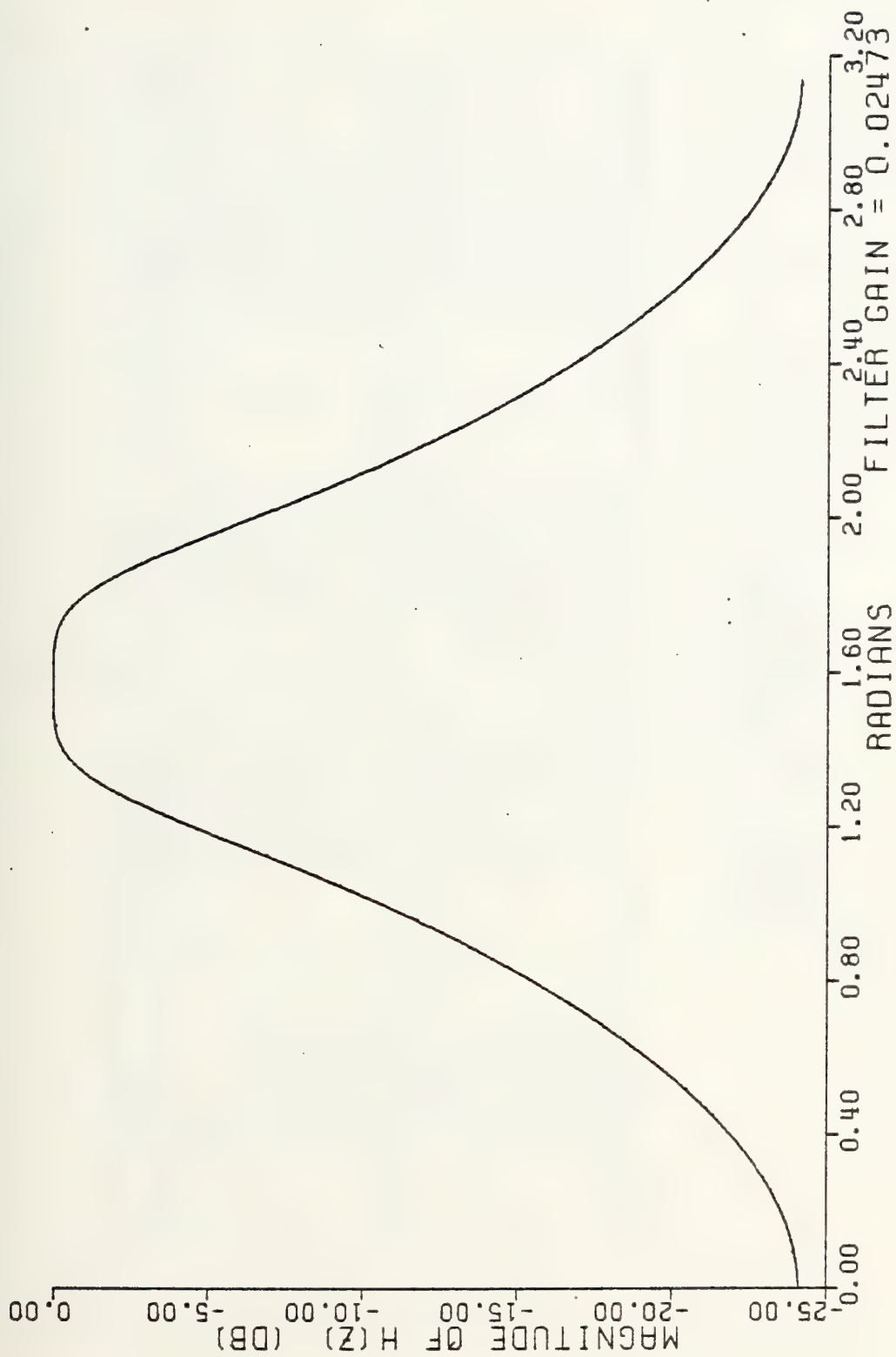


Figure 7-2g Magnitude Of  $H(z)$  For A Fourth-Order Band-Pass Filter









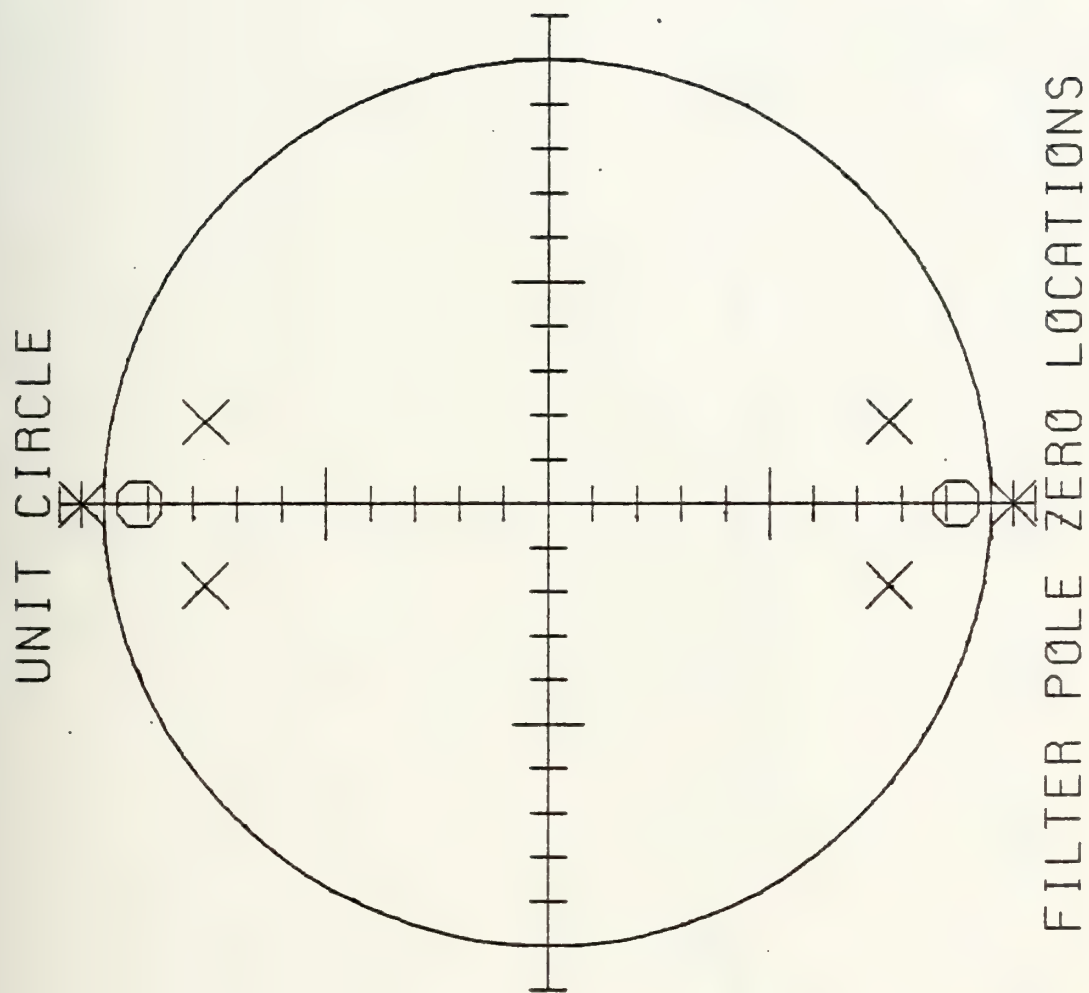


Figure 7-3b Plot Of Pole Zero Locations For A Fourth-Order Notch Filter



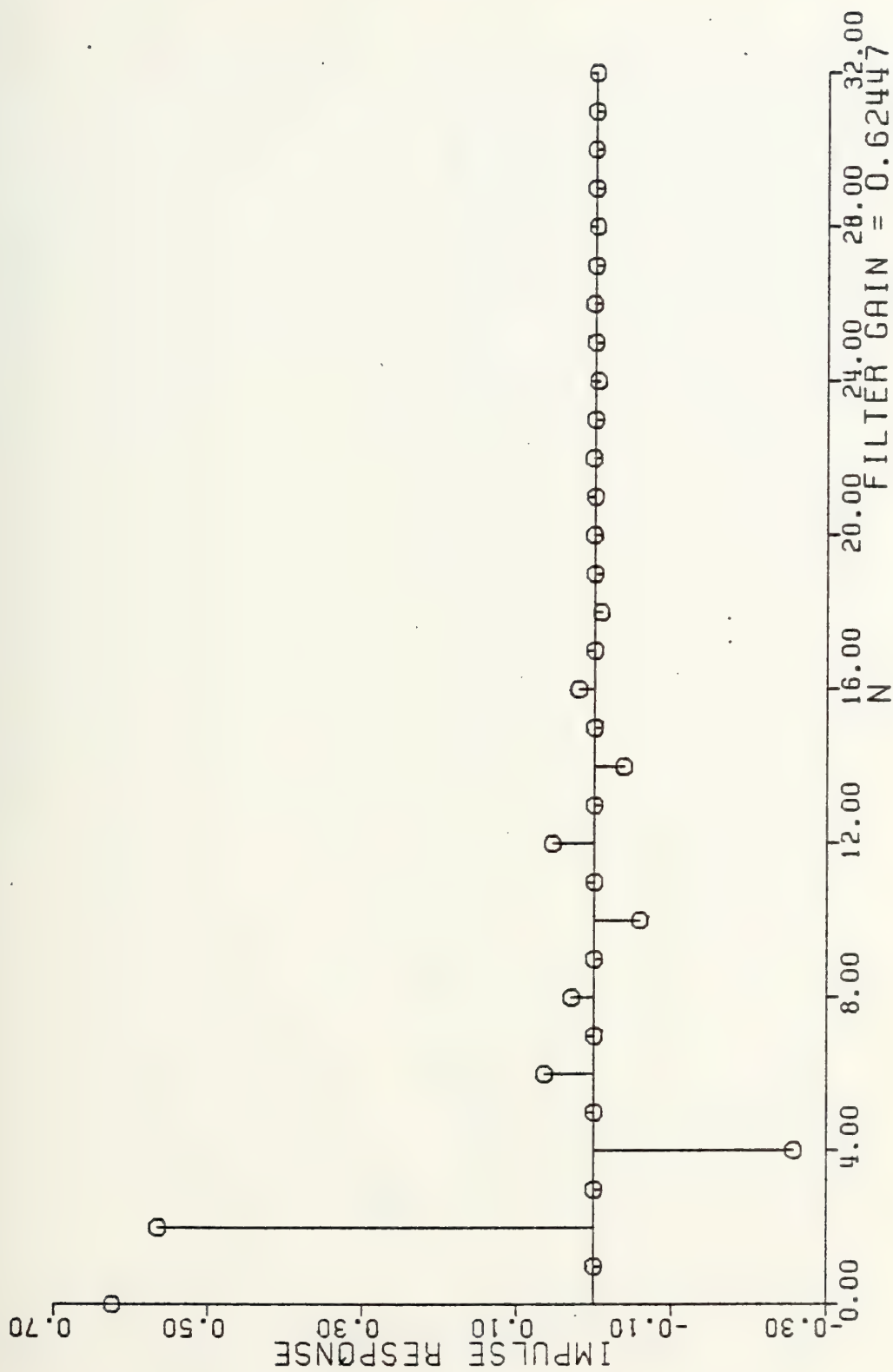


Figure 7-3c Unit Sample Response For A Fourth-Order Notch Filter



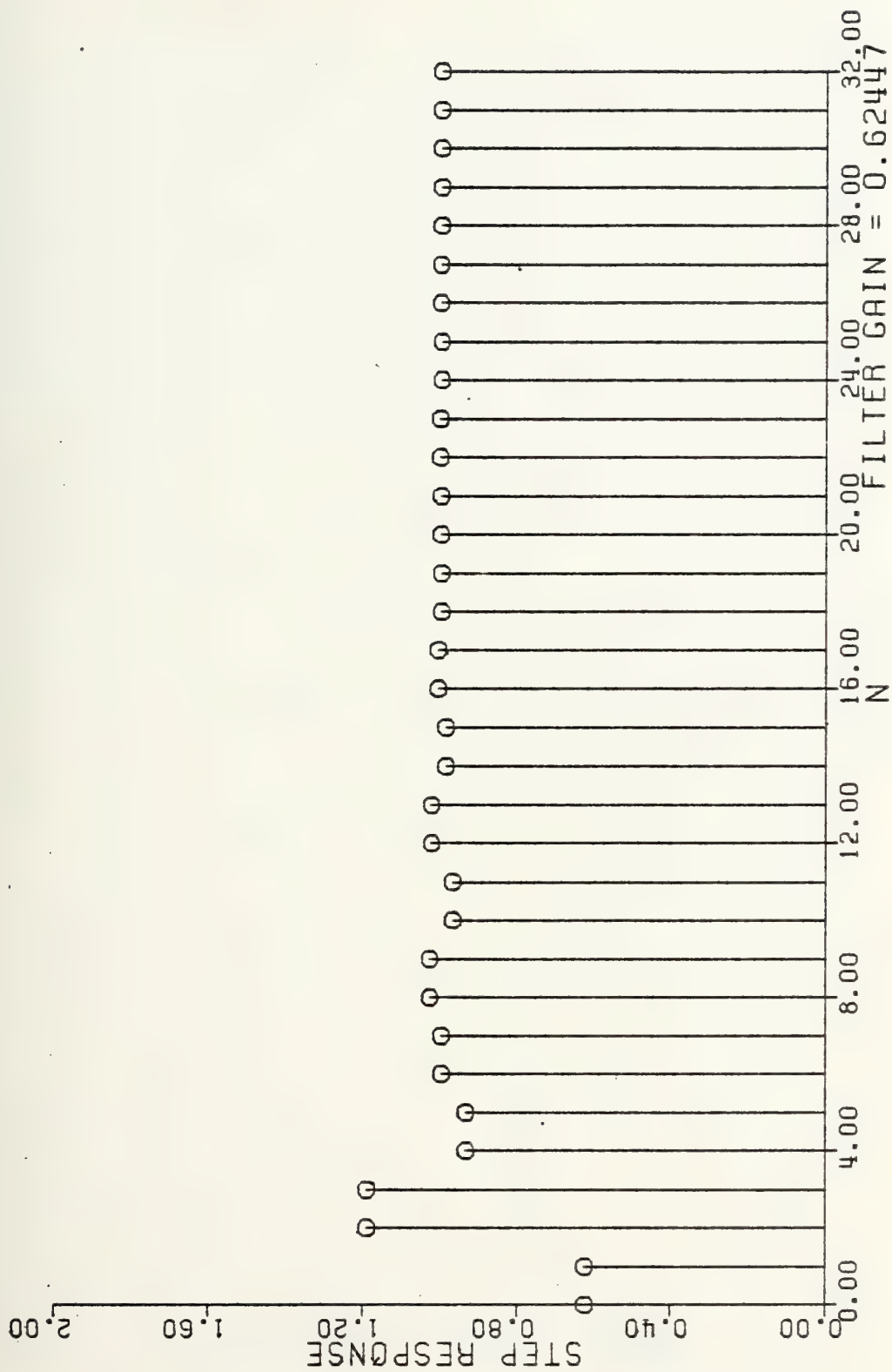


Figure 7-3d Unit Step Response For A Fourth-Order Notch Filter





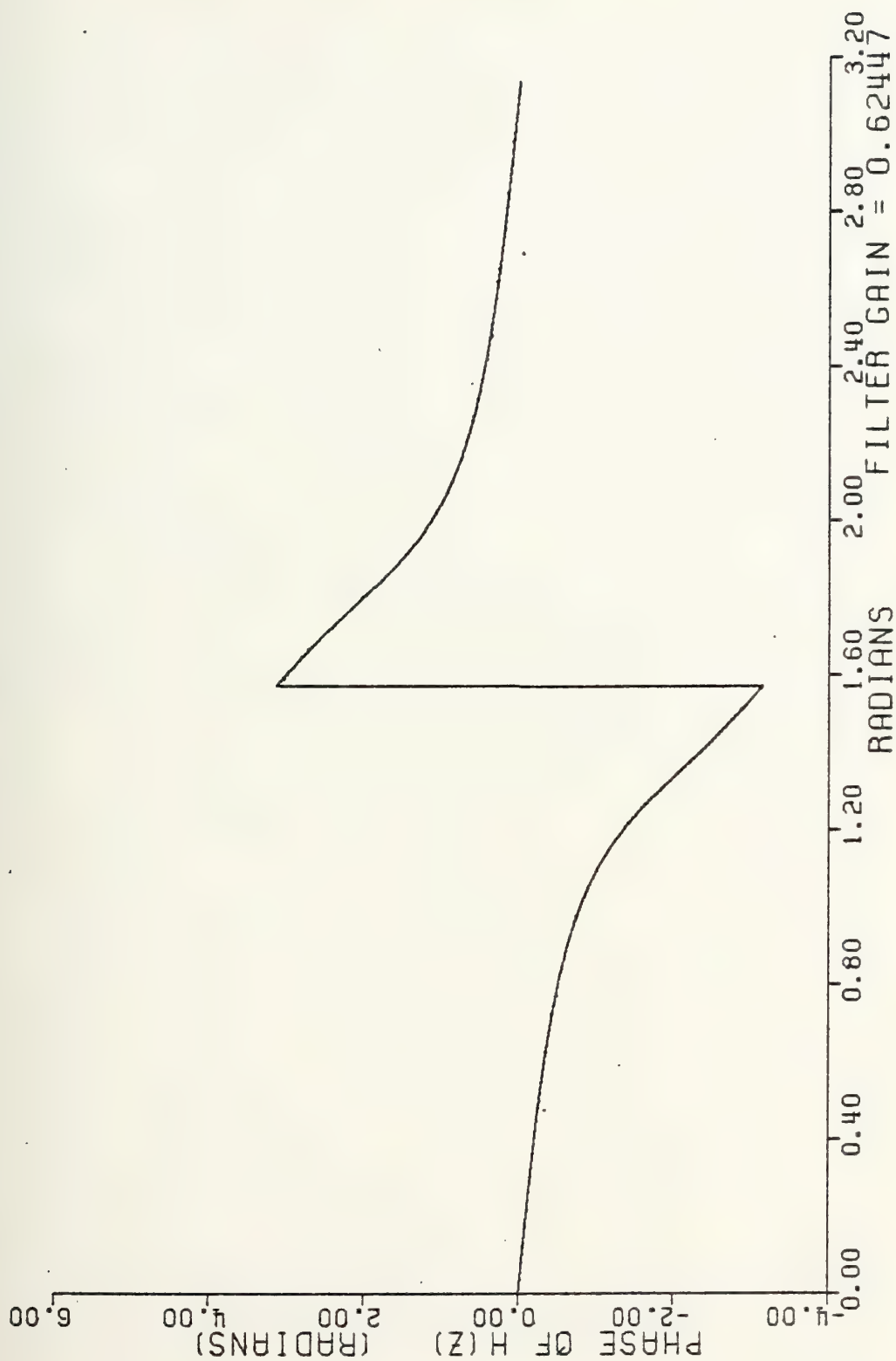


Figure 7-3e Phase Response For a Fourth-Order Notch Filter



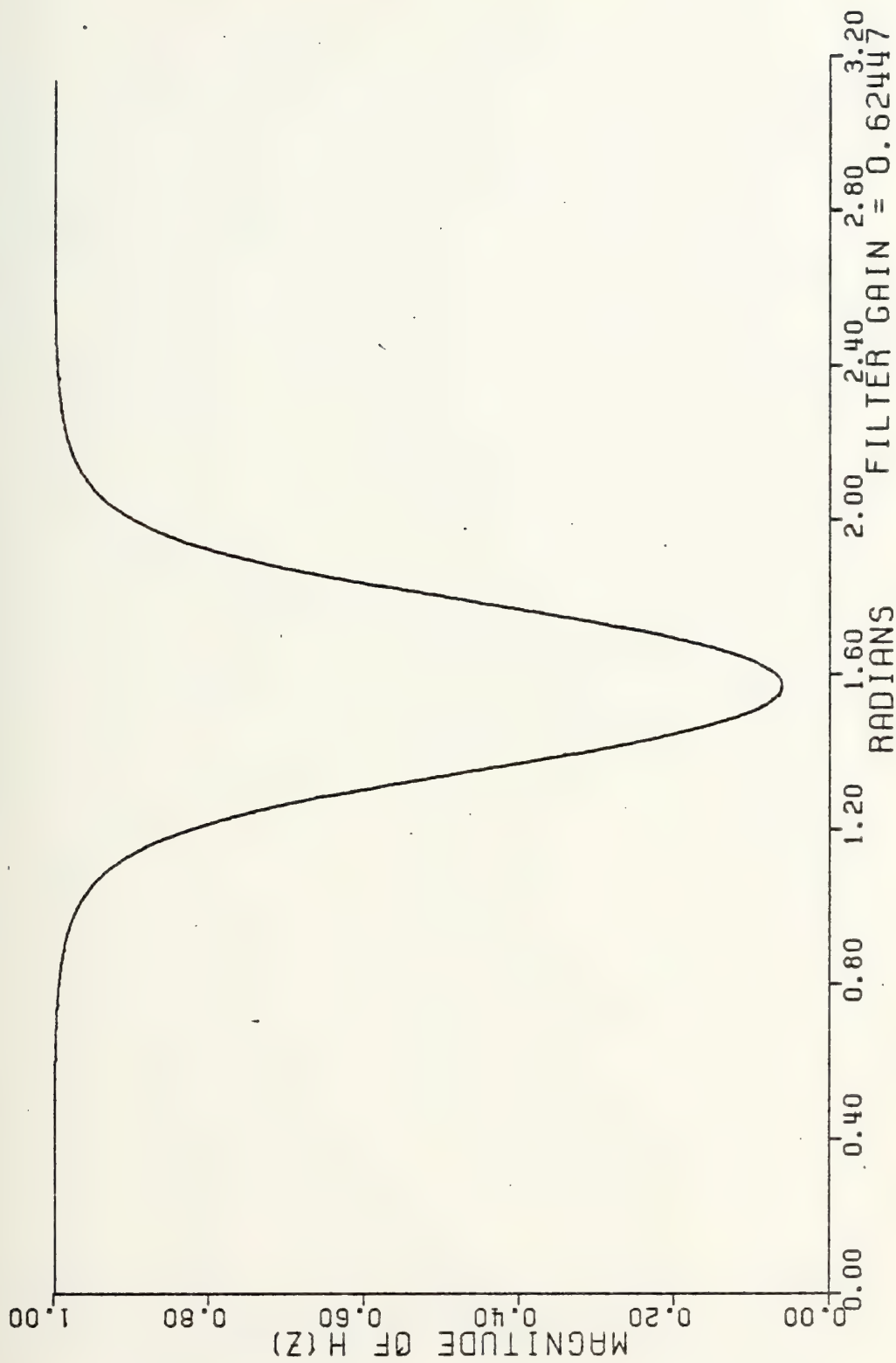


Figure 7-3f Magnitude Of  $H(z)$  For A Fourth-Order Notch Filter



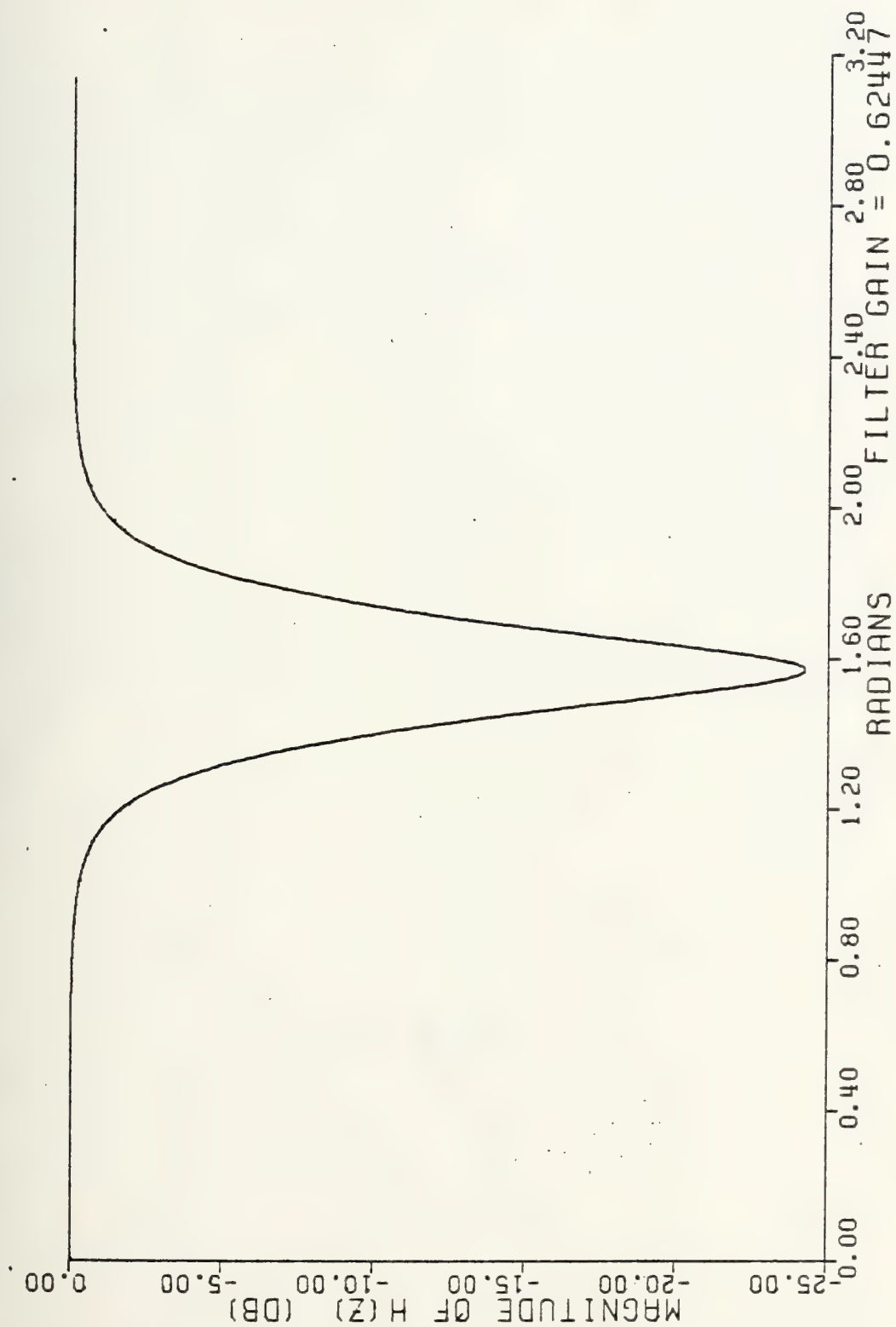


Figure 7-3g Magnitude In Decibels For A Fourth-Order Notch Filter



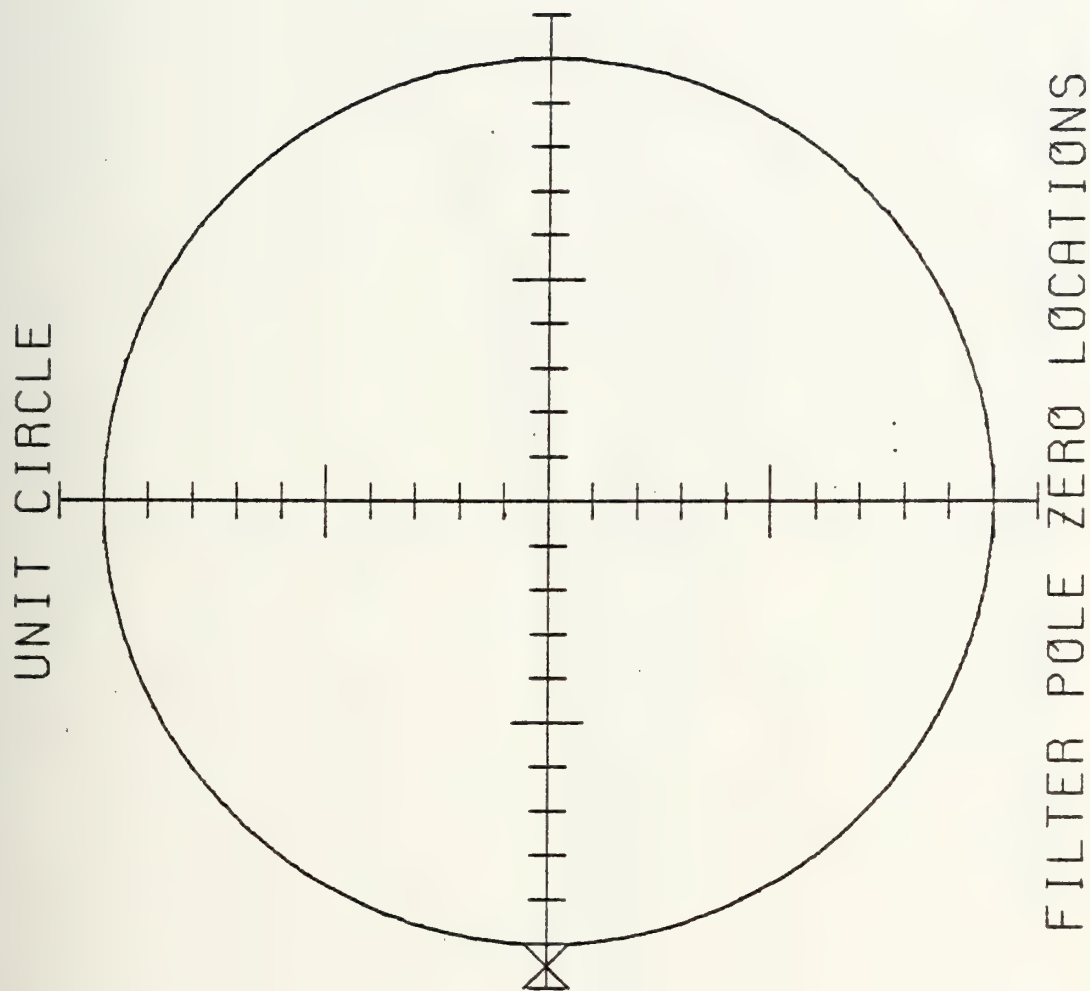


Figure 7-4a Example Of A Single Real Pole Outside The Unit Circle At  $(-1.05)$





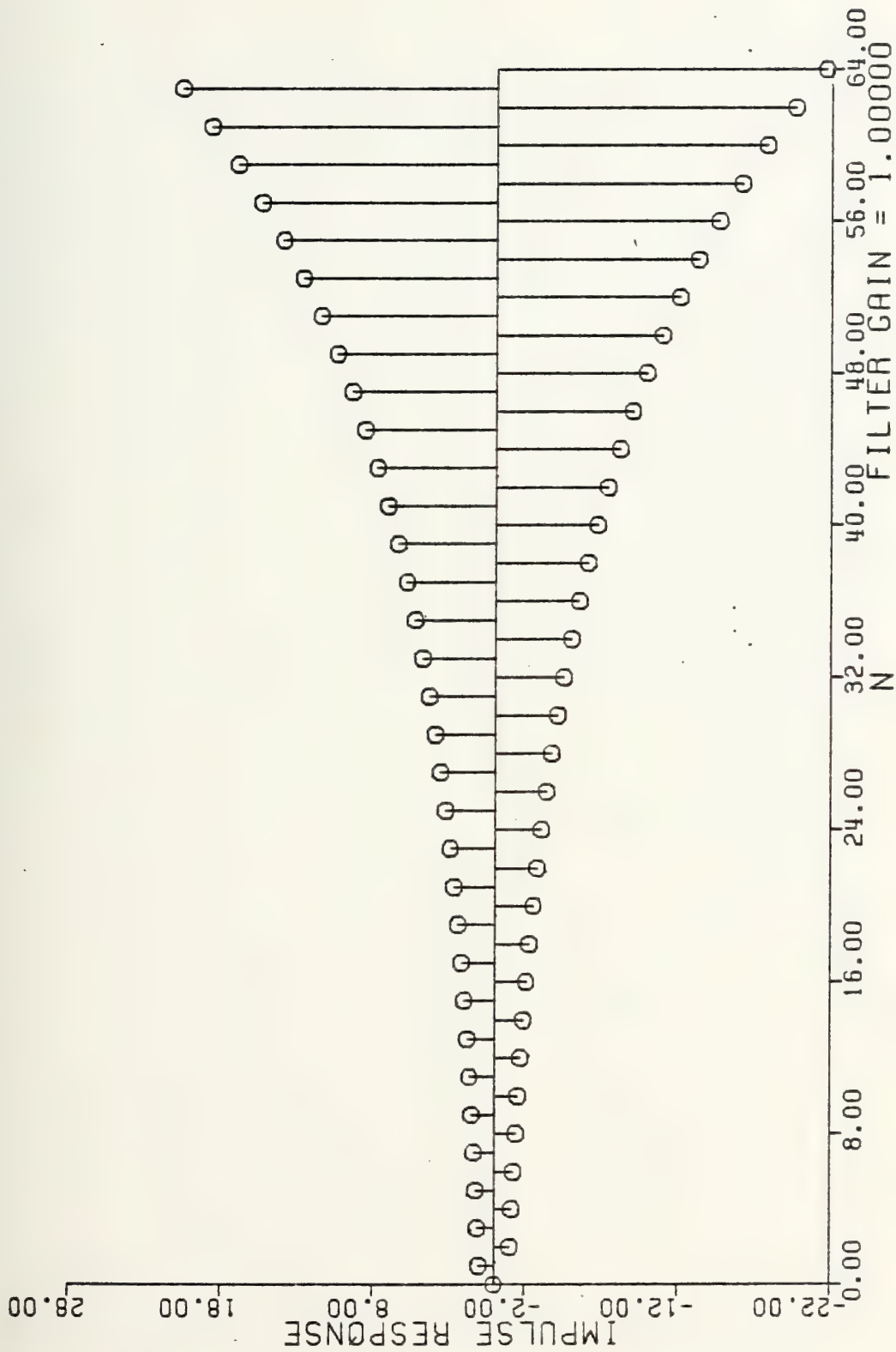


Figure 7-4b Unit Sample Response For A Single Real Pole At  $(-1.05)$



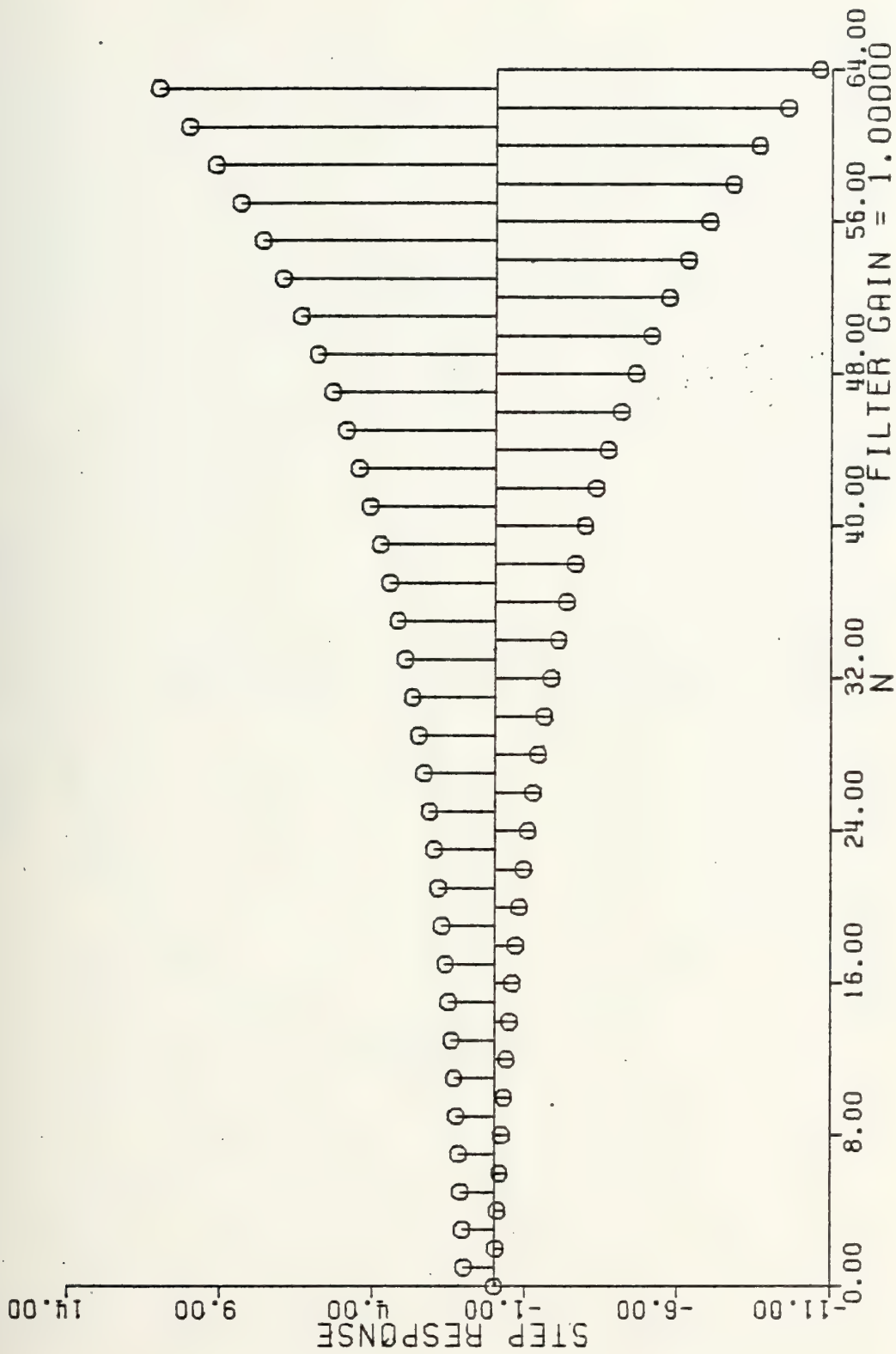


Figure 7-4c Unit Step Response For A Single Real Pole At (-1.05)



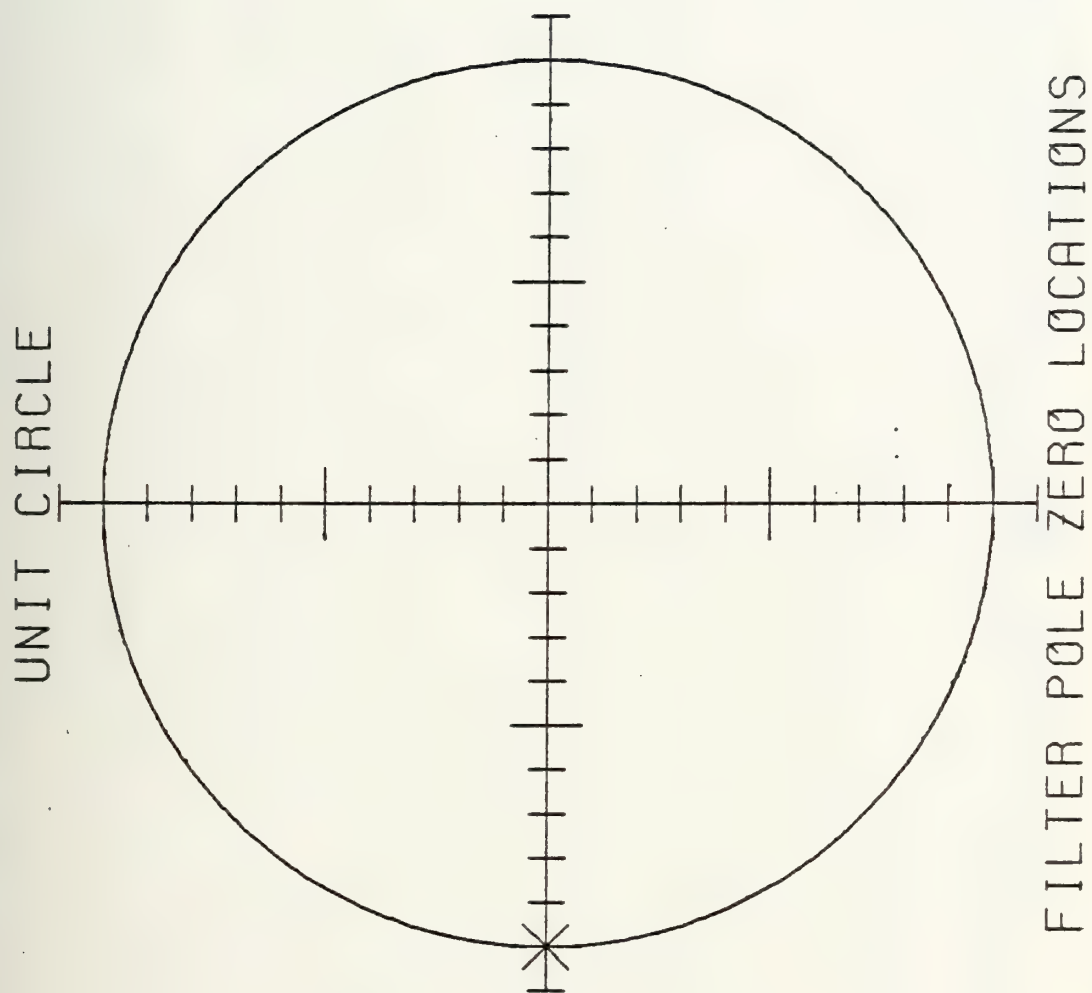


Figure 7-5a Example Of A Single Real Pole On The Unit Circle At  $(-1.0)$



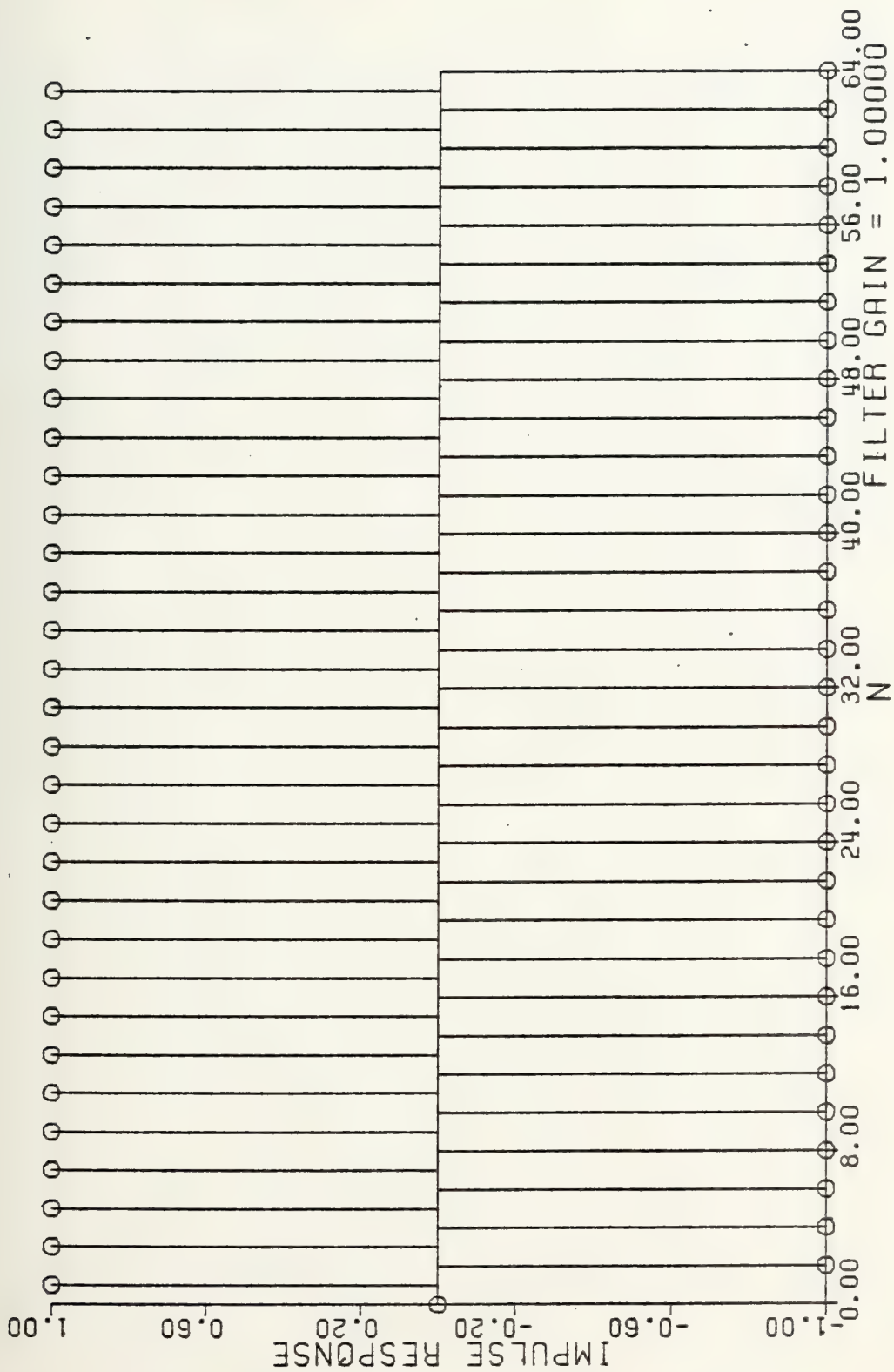


Figure 7-5b Unit Sample Response For A Single Real Pole At (-1.0)





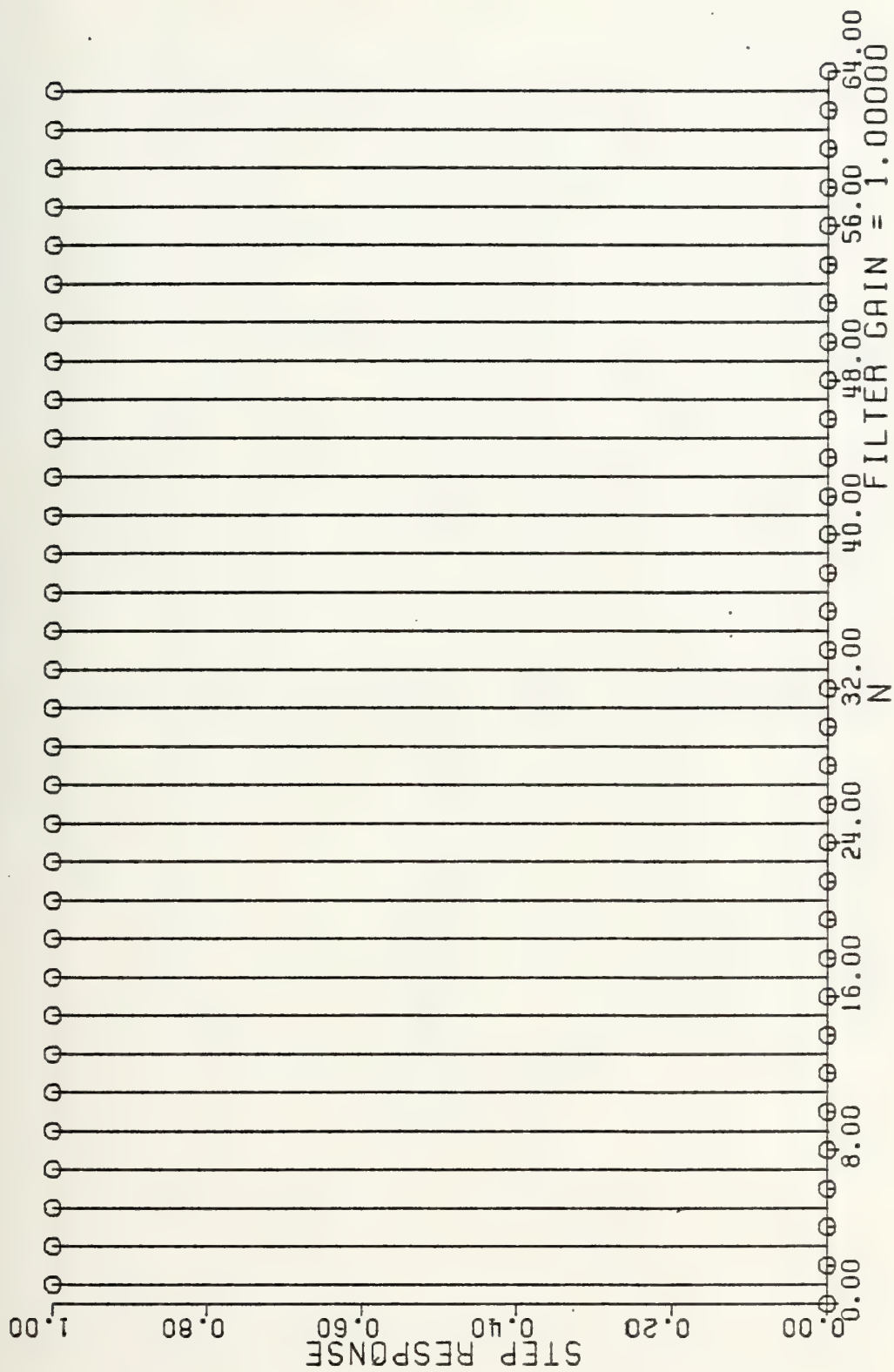


Figure 7-5c Unit Step Response For A Single Real Pole At (-1.0)



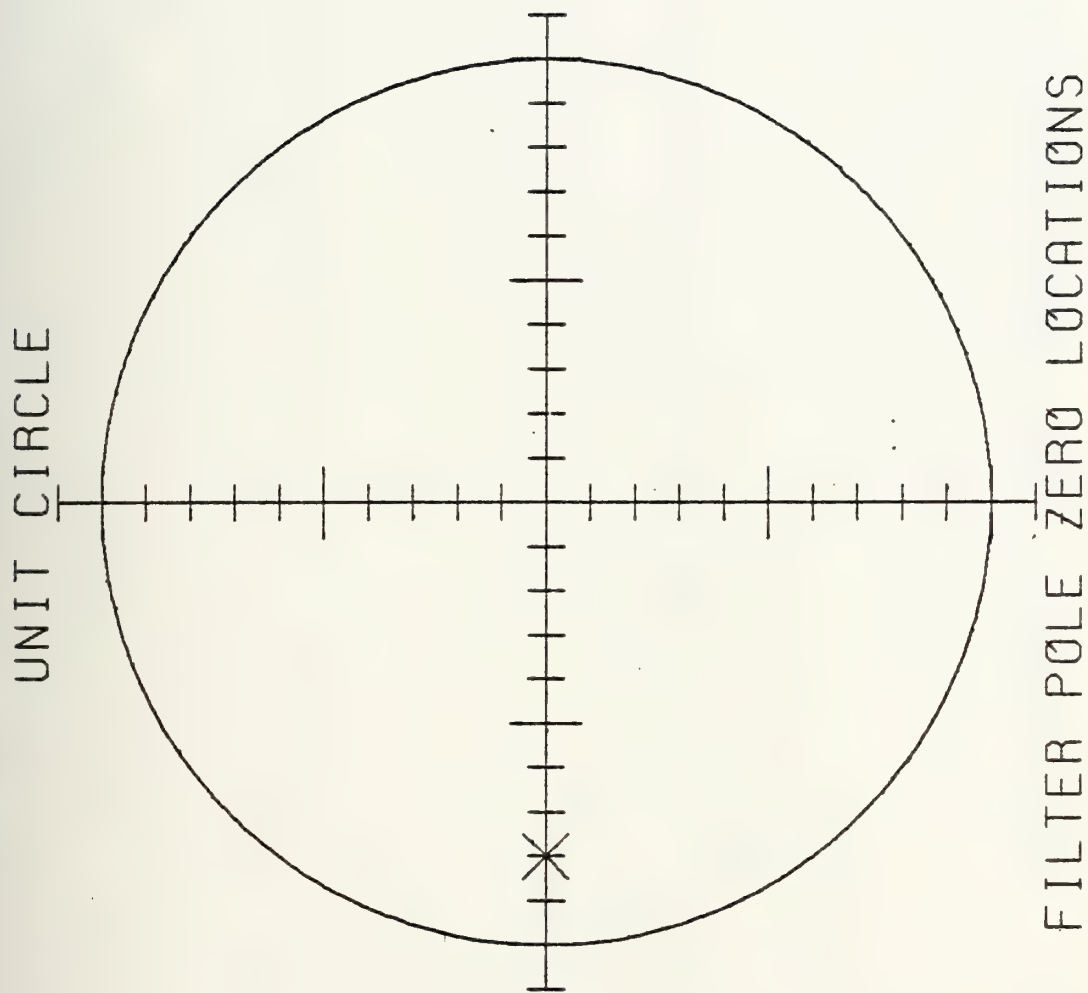


Figure 7-6a Example Of A Single Real Pole At  $(-0.8)$



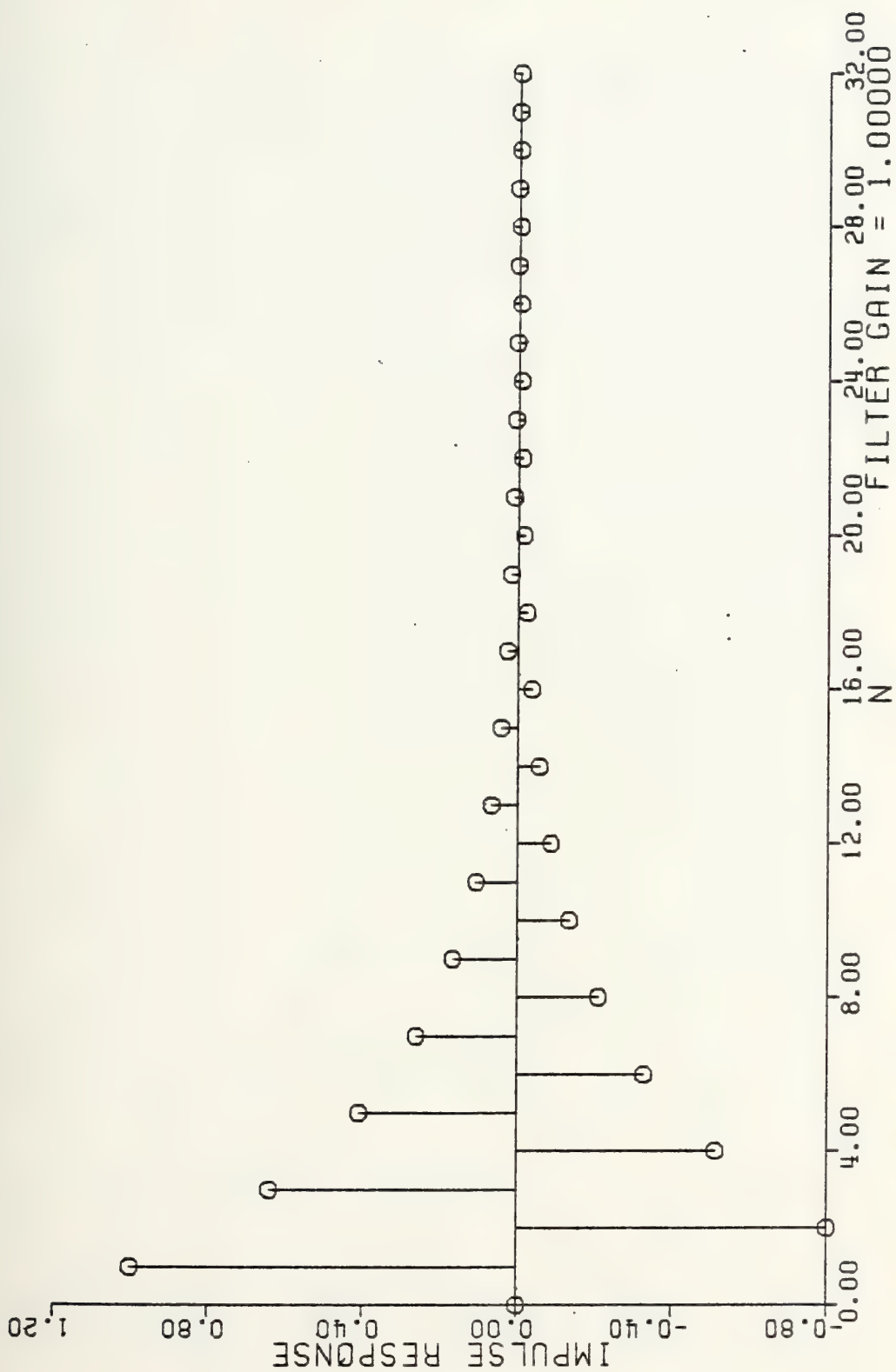


Figure 7-6b Unit Sample Response For A Single Real Pole at (-1.0)



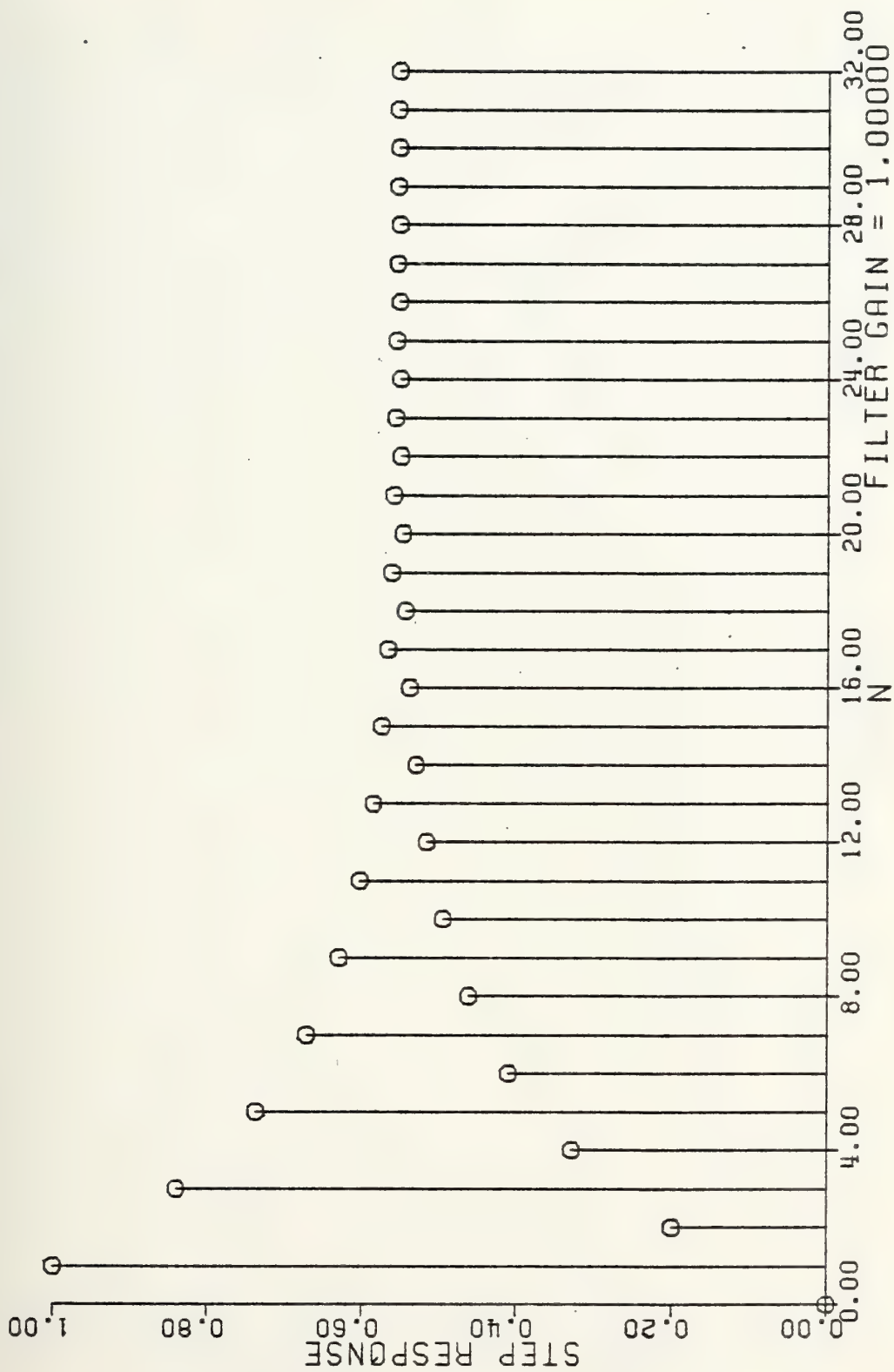


Figure 7-6c Unit Step Response For A Single Real Pole At  $(-0.8)$





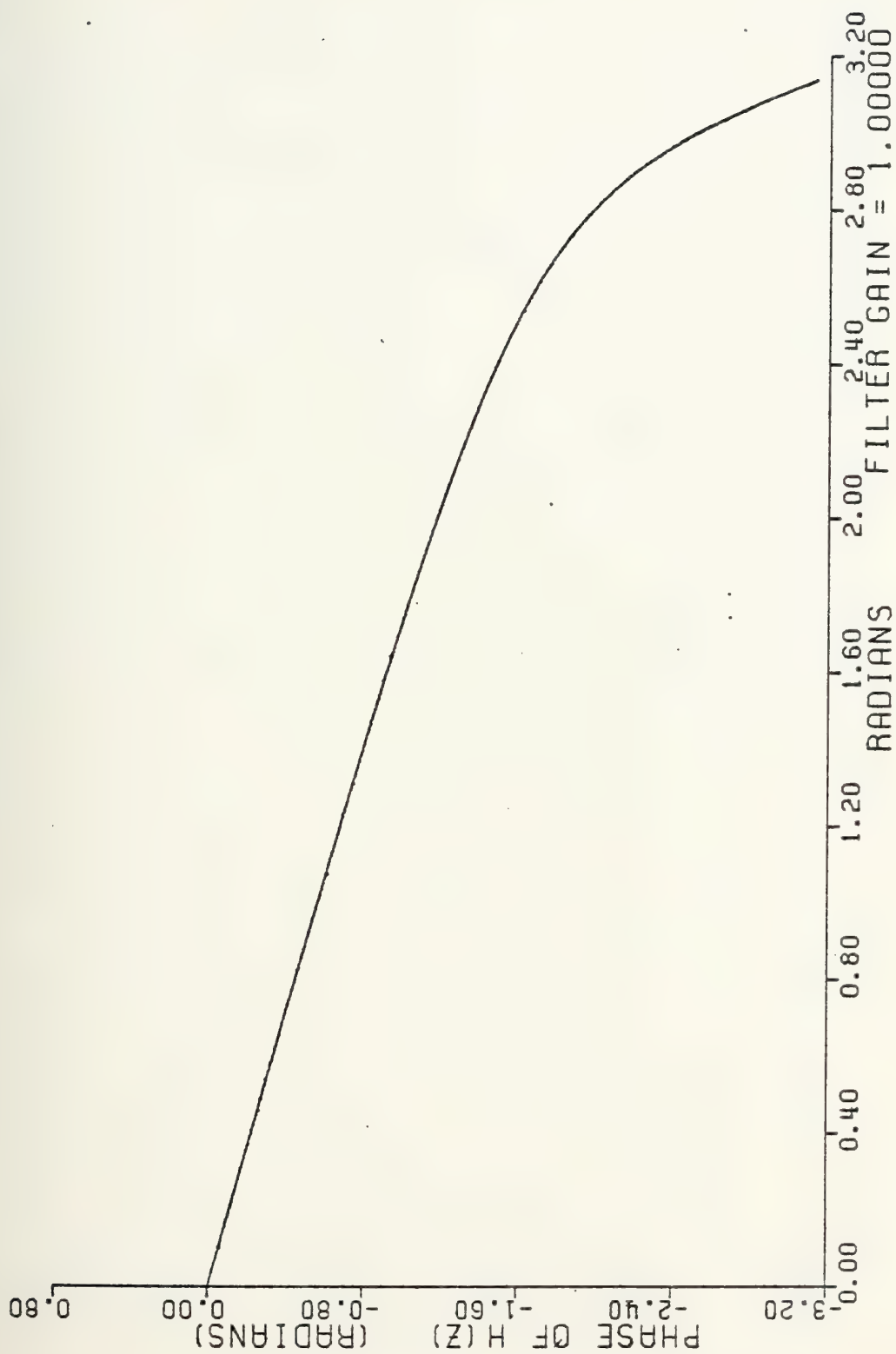


Figure 7-7d Phase Response For A Single Real Pole At  $(-0.8)$



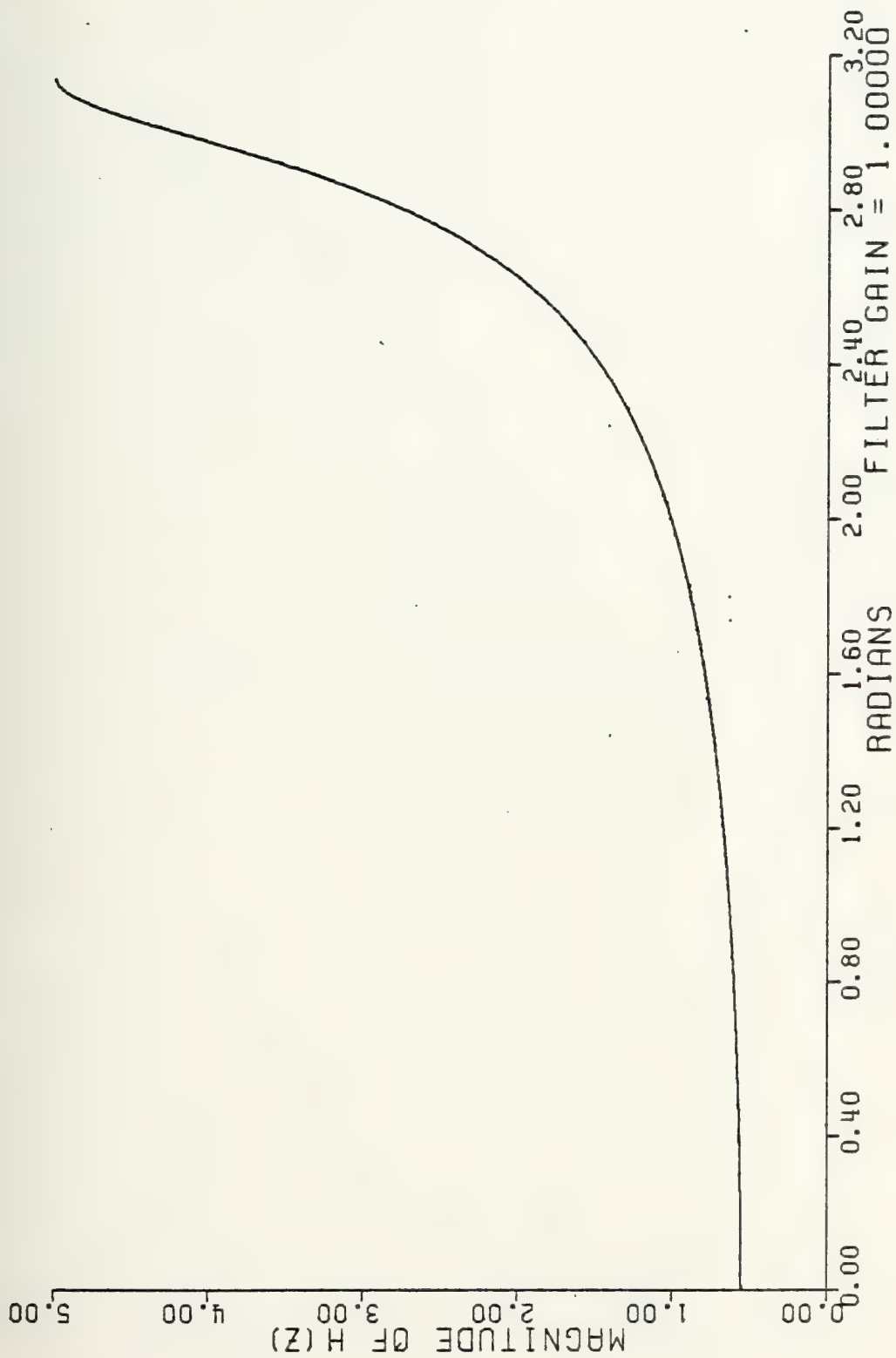


Figure 7-6e Magnitude of  $H(z)$  For A Single Real Pole At  $(-0.8)$



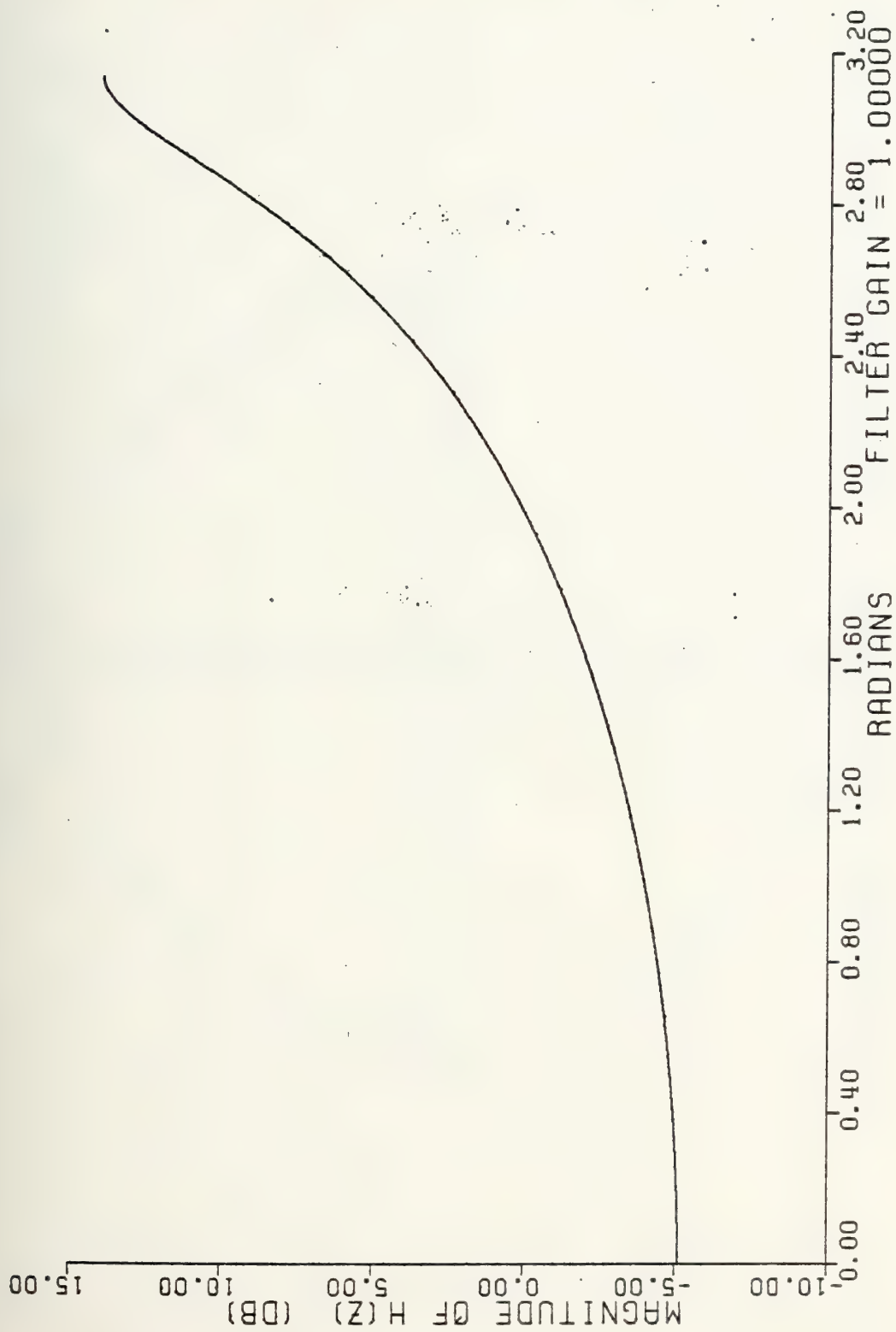


Figure 7-6f Magnitude Of  $H(z)$  In Decibels For a Single Real Pole at  $(-0.8)$



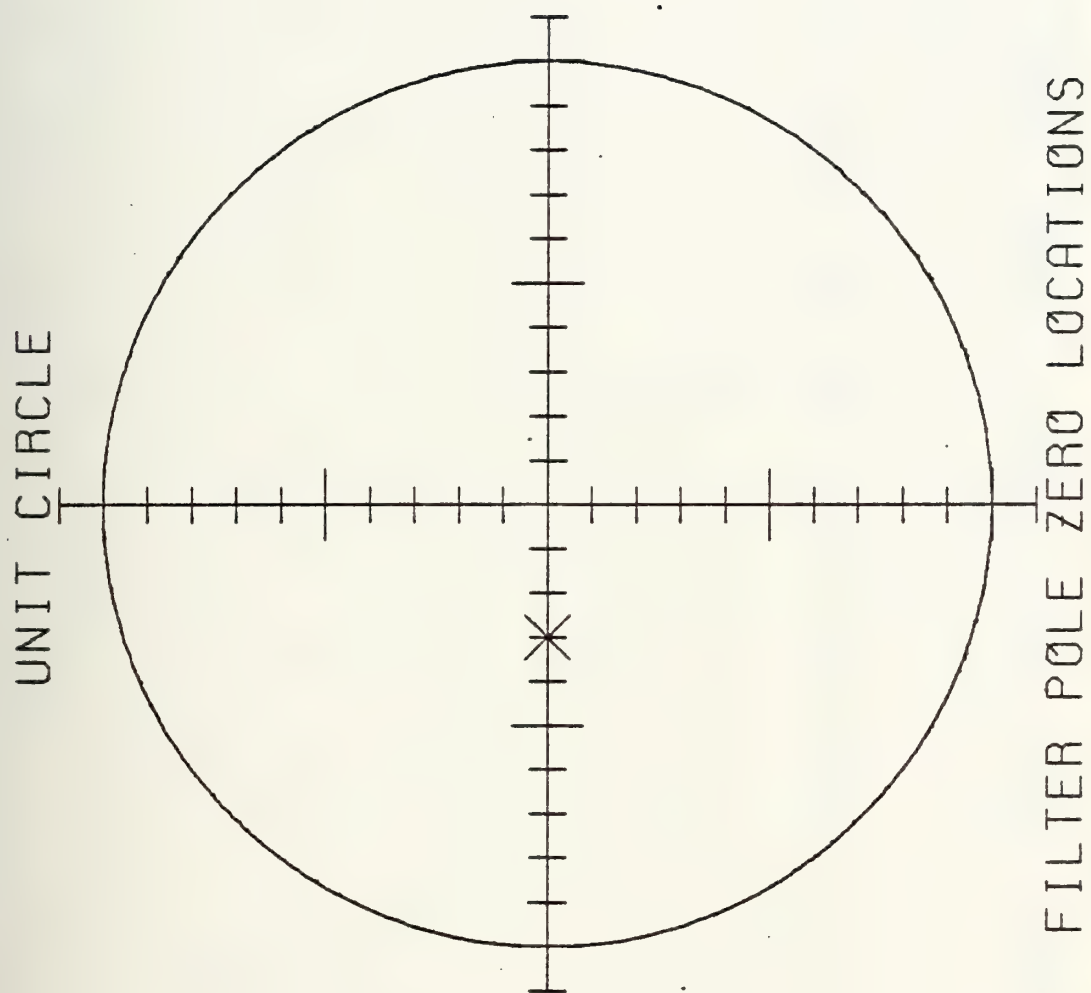


Figure 7-7a Example Of A Single Real Pole At (-0.3)





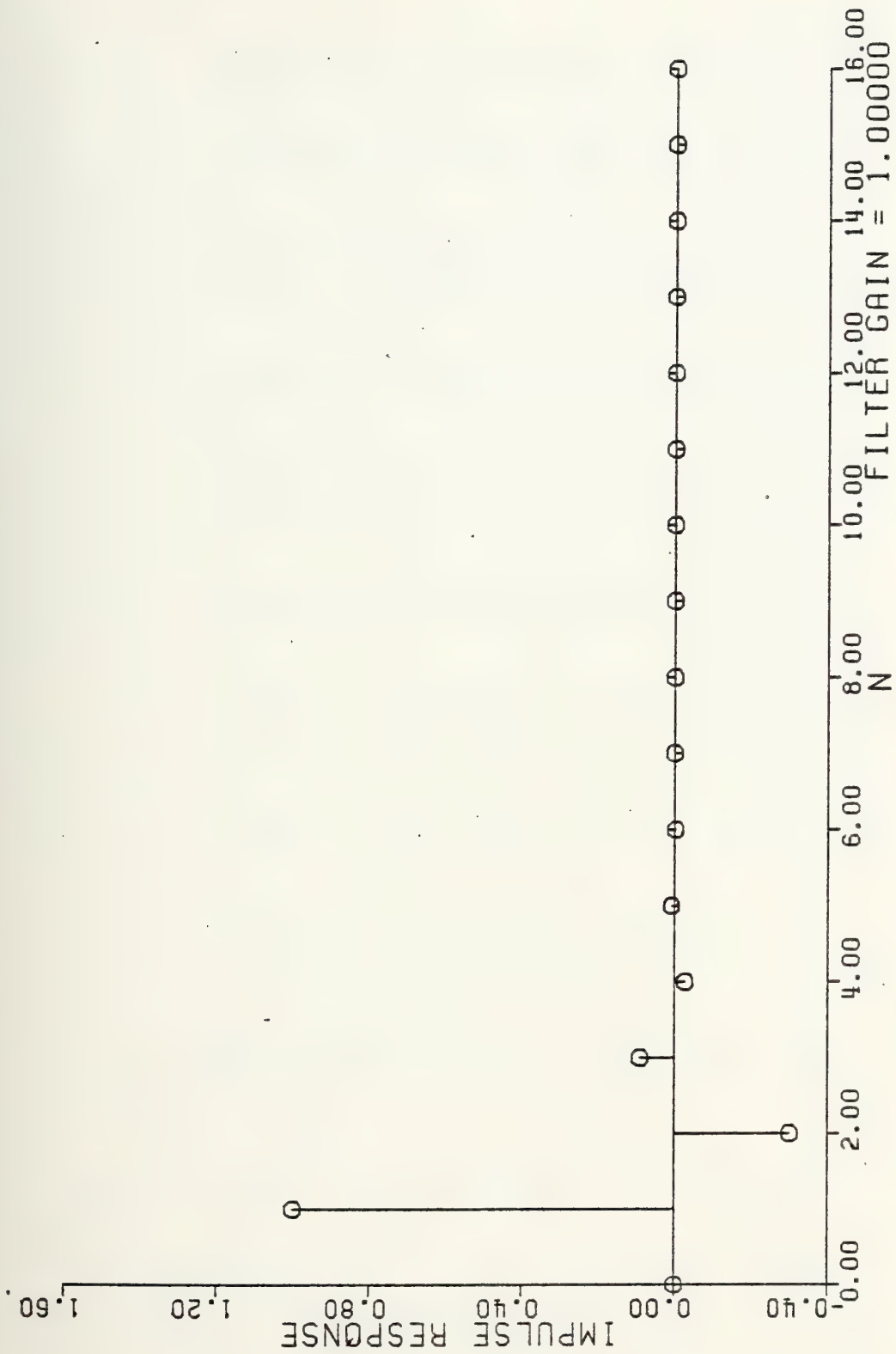


Figure 7-7b Unit Sample Response For A Single Real Pole At (-0.3)



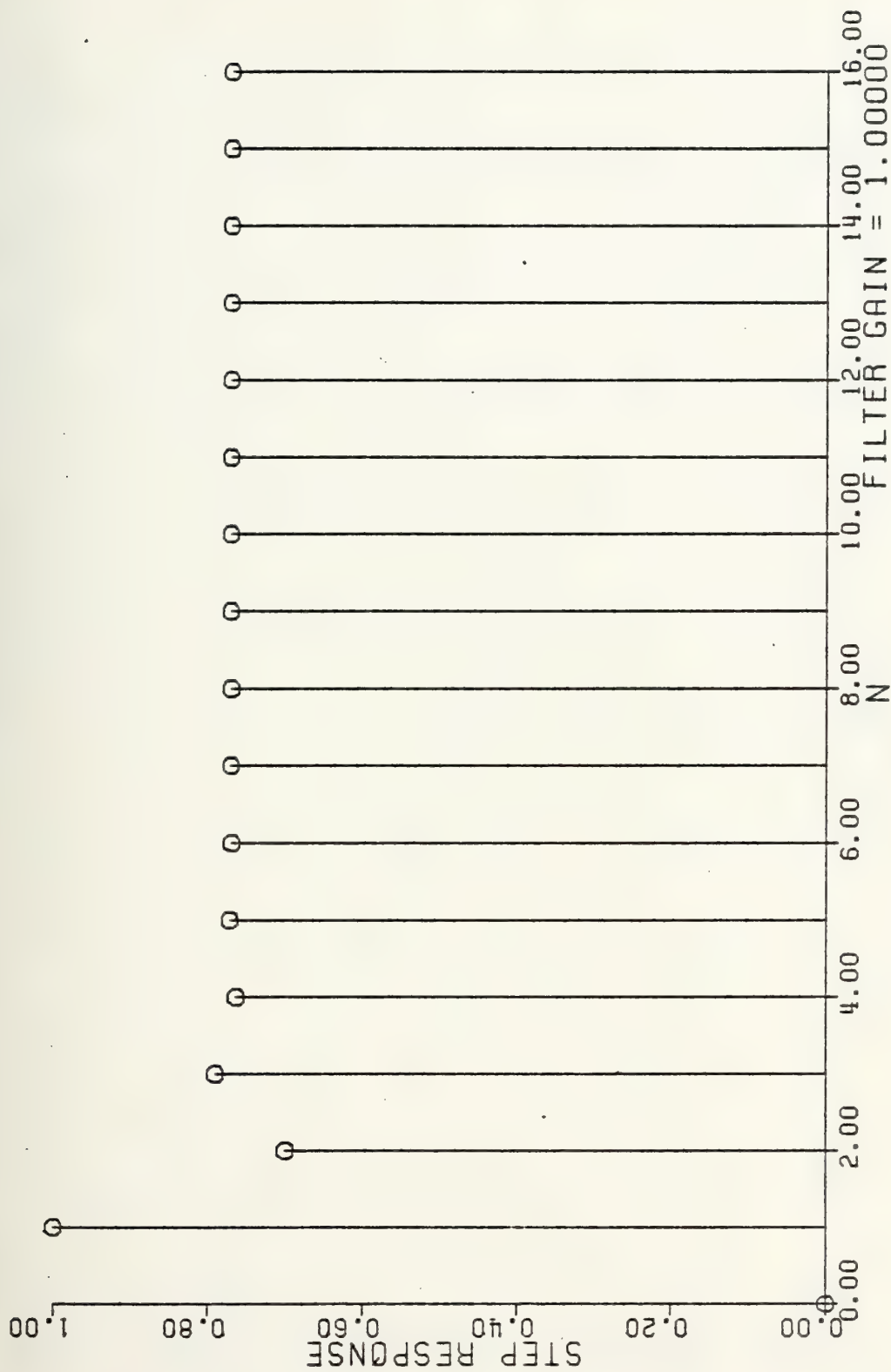


Figure 7-7c Unit Step Response For A Single Real Pole At  $(-0.3)$



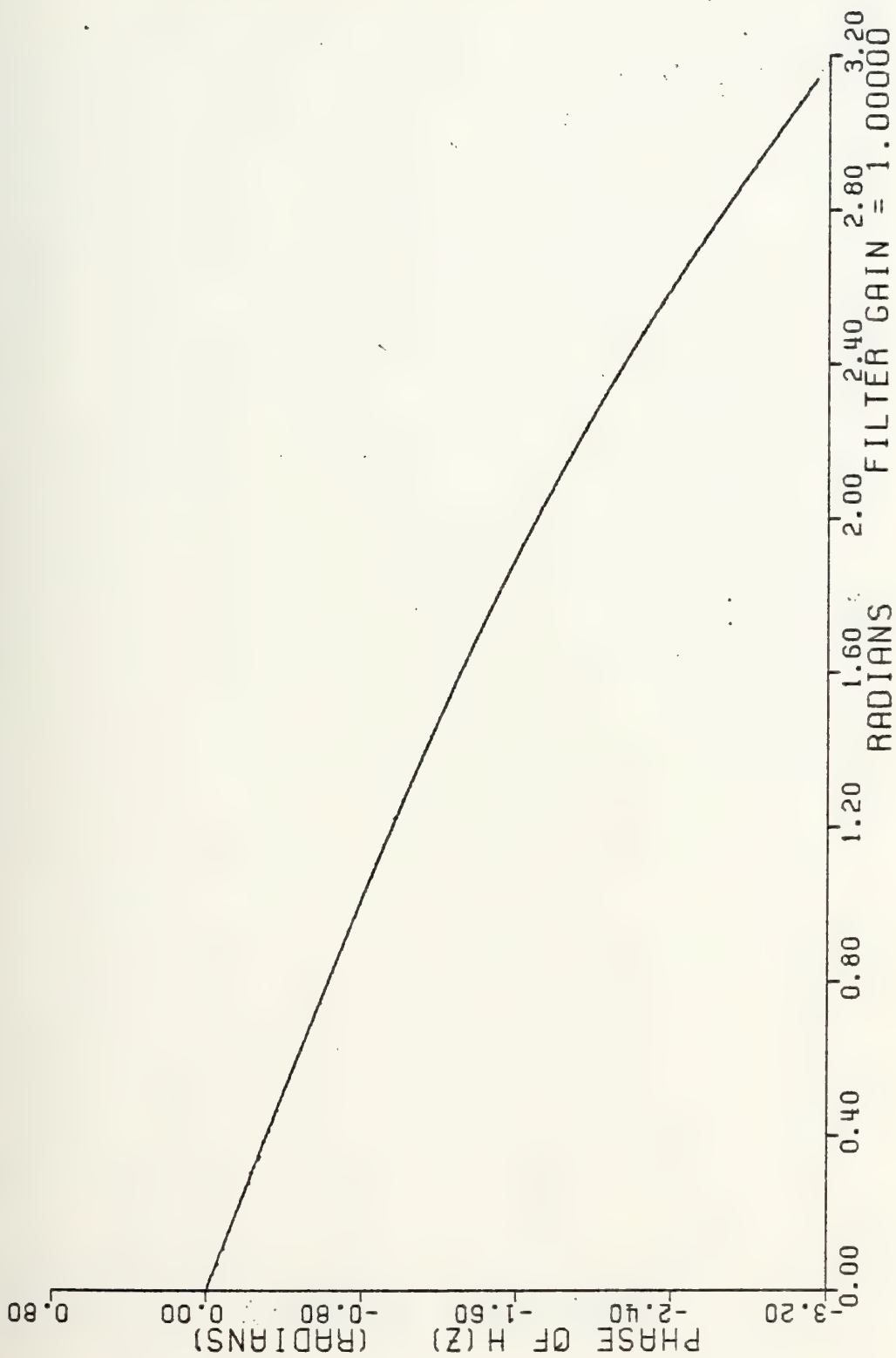


Figure 7-7d Phase Response For A Single Real Pole At  $(-0.3)$



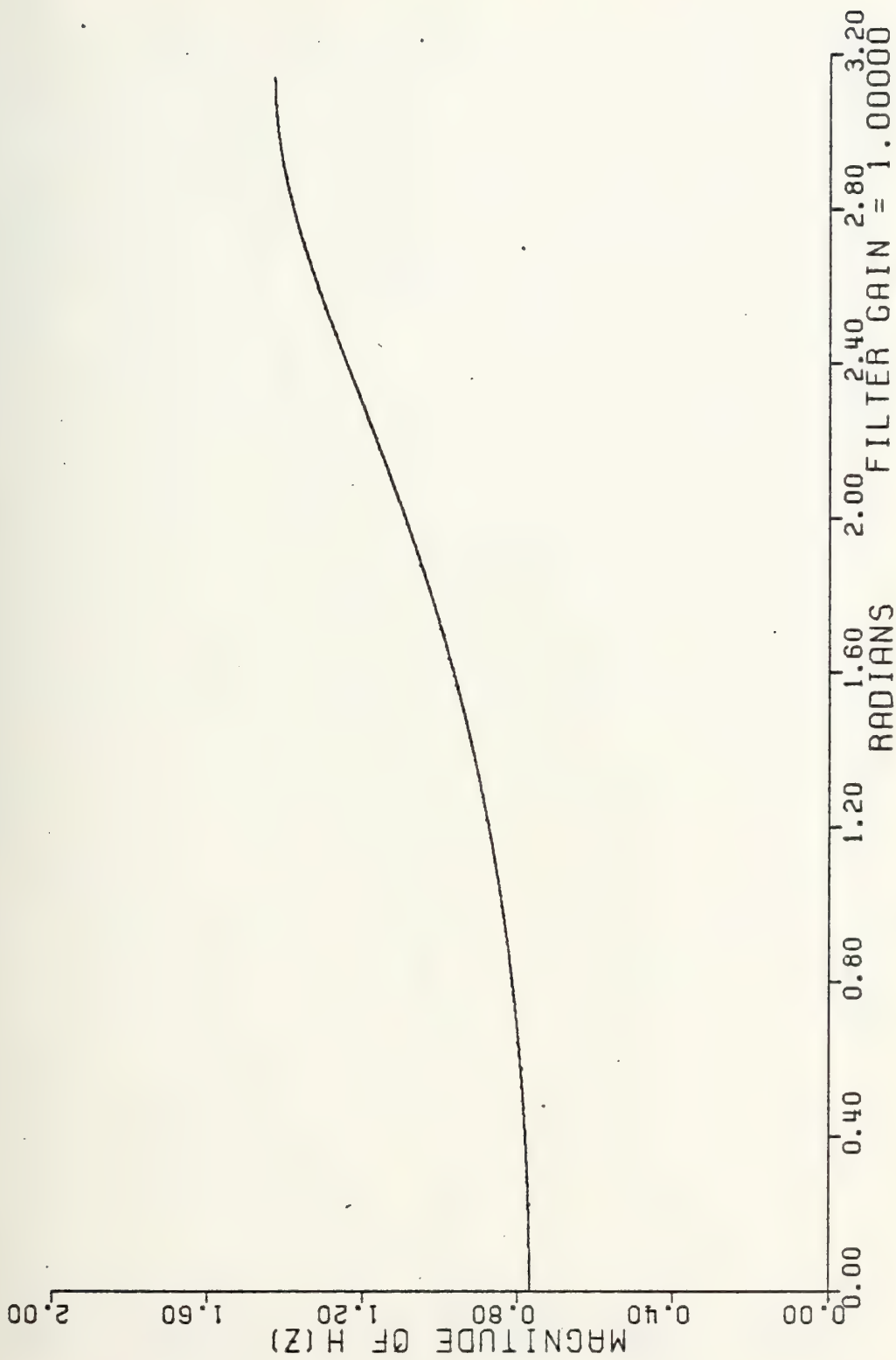


Figure 7-7e Magnitude Of  $H(z)$  For A Single Real Pole At  $(-0.3)$





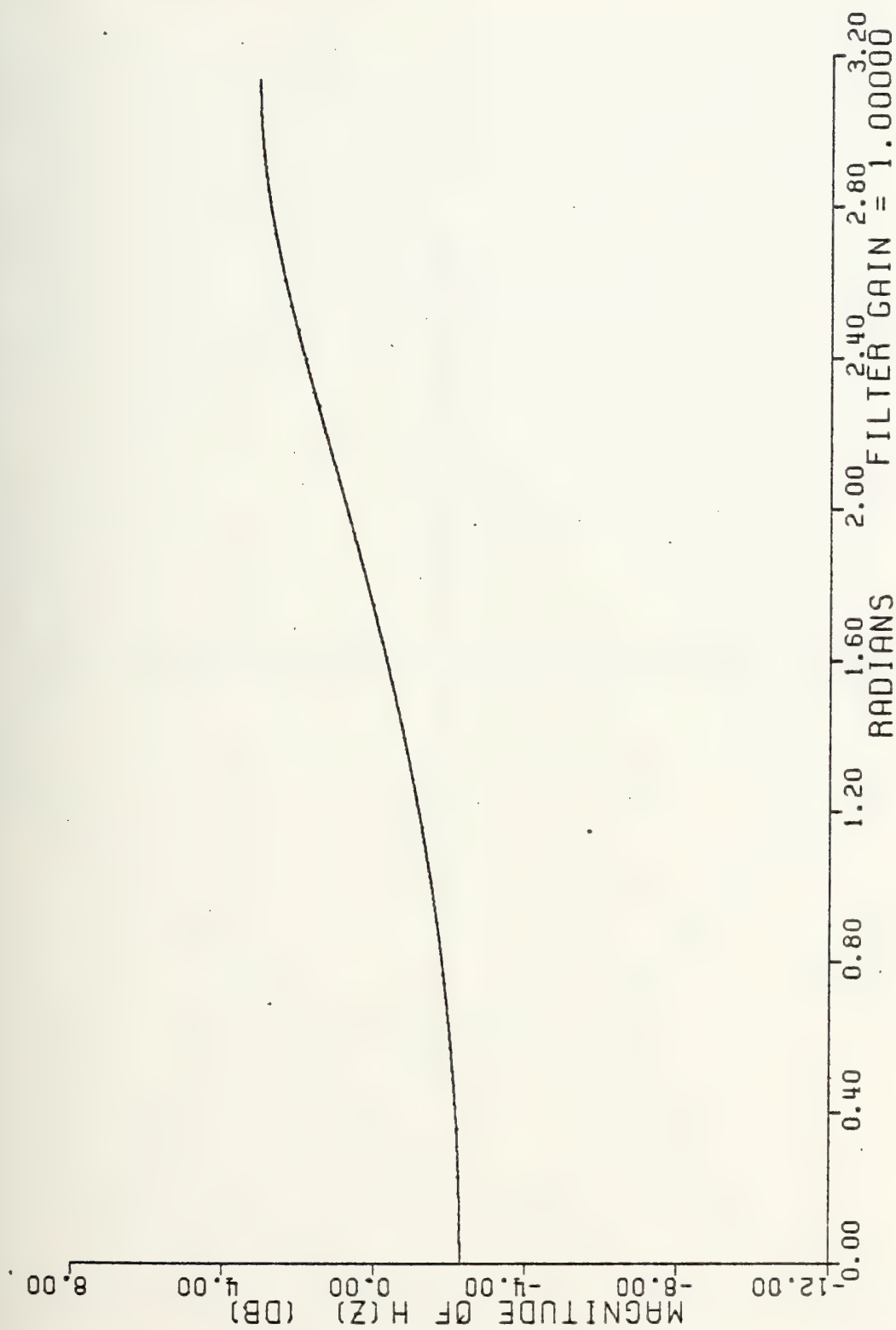


Figure 7-7f Magnitude Of  $H(z)$  For A Single Real Pole At  $(-0.3)$



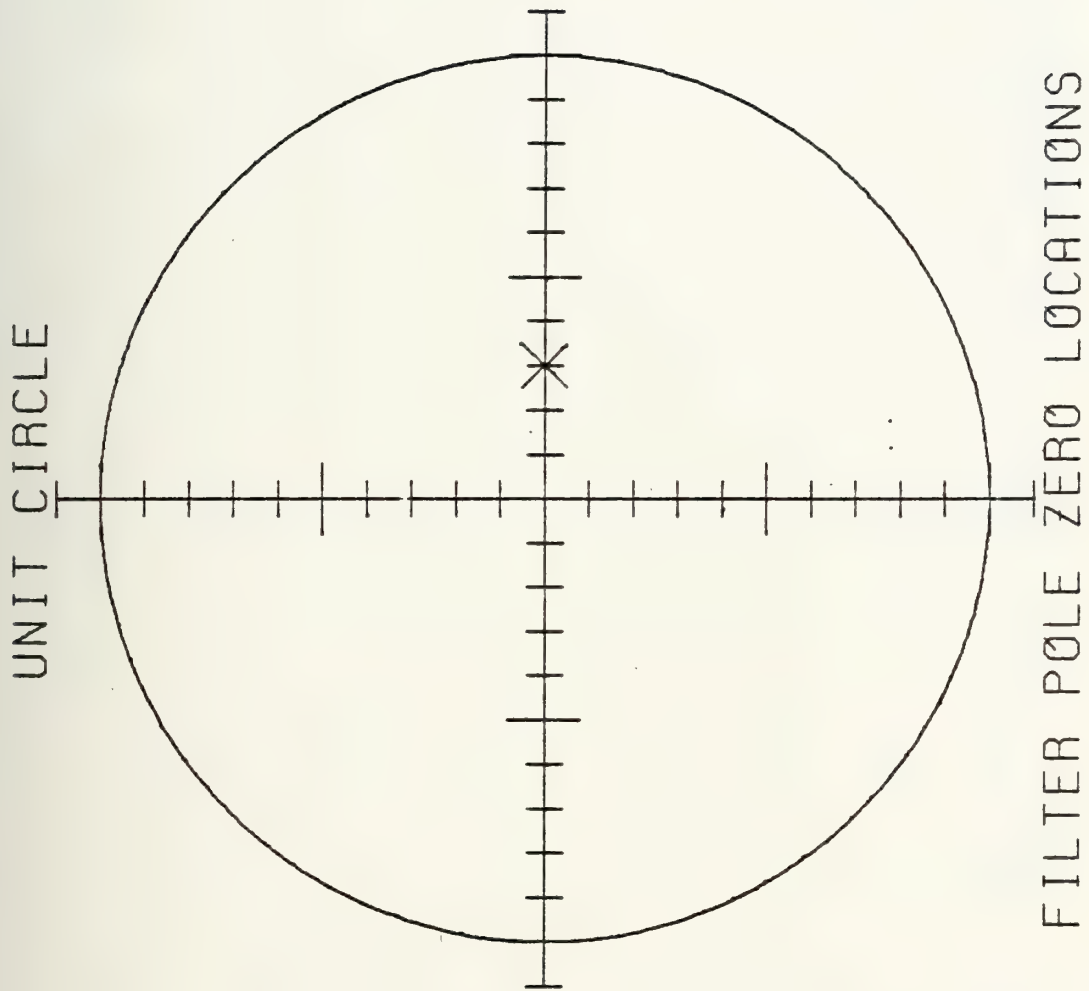


Figure 7-8a Example Of A Single Real Pole At (0.3)



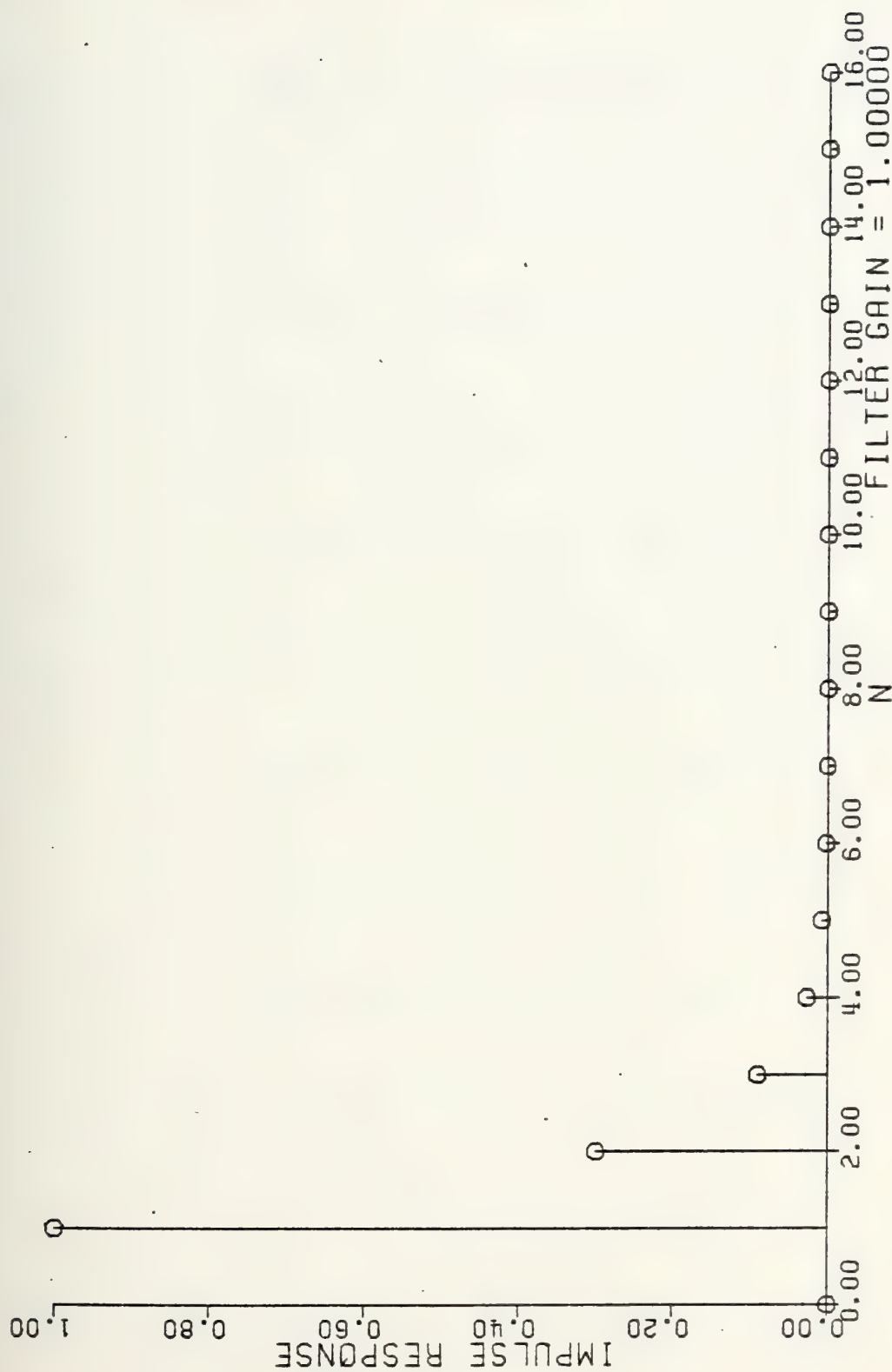


Figure 7-8b Unit Sample Response For A Single Real Pole At (0.3)



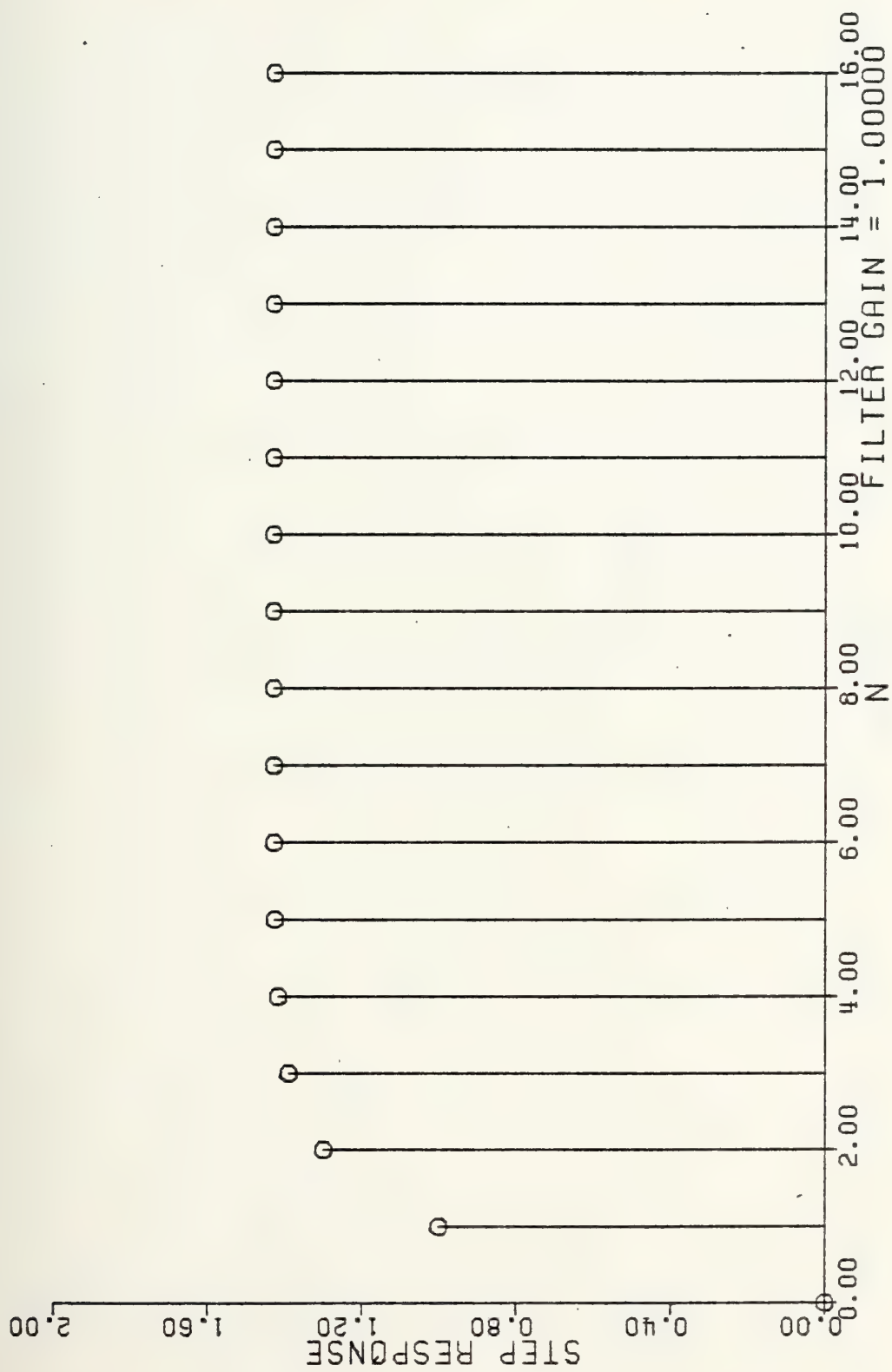


Figure 7-8c Unit Step Response For A Single Real Pole At (0.3)





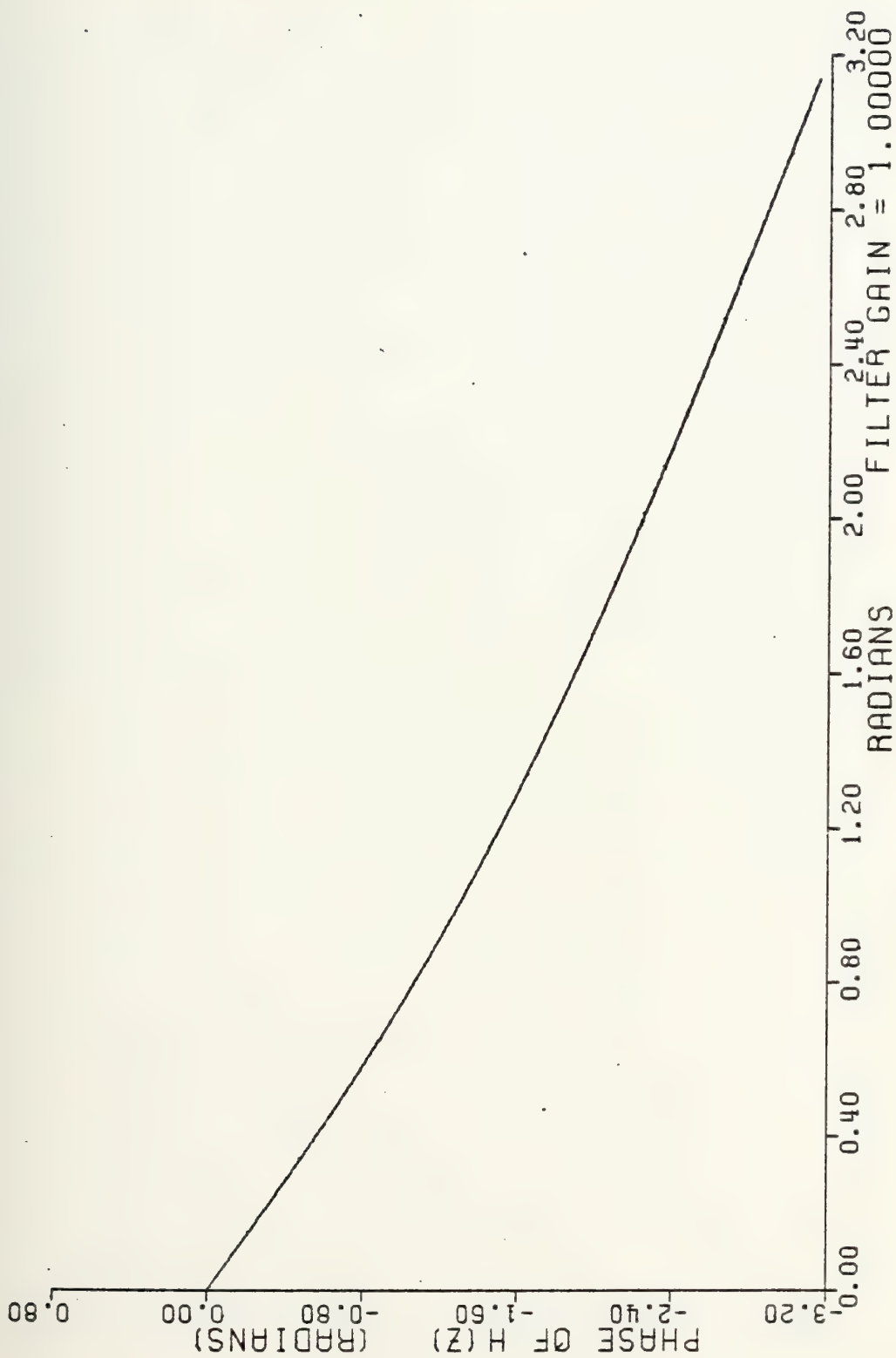


Figure 7-8d Phase Response For A Single Real Pole At (0.3)



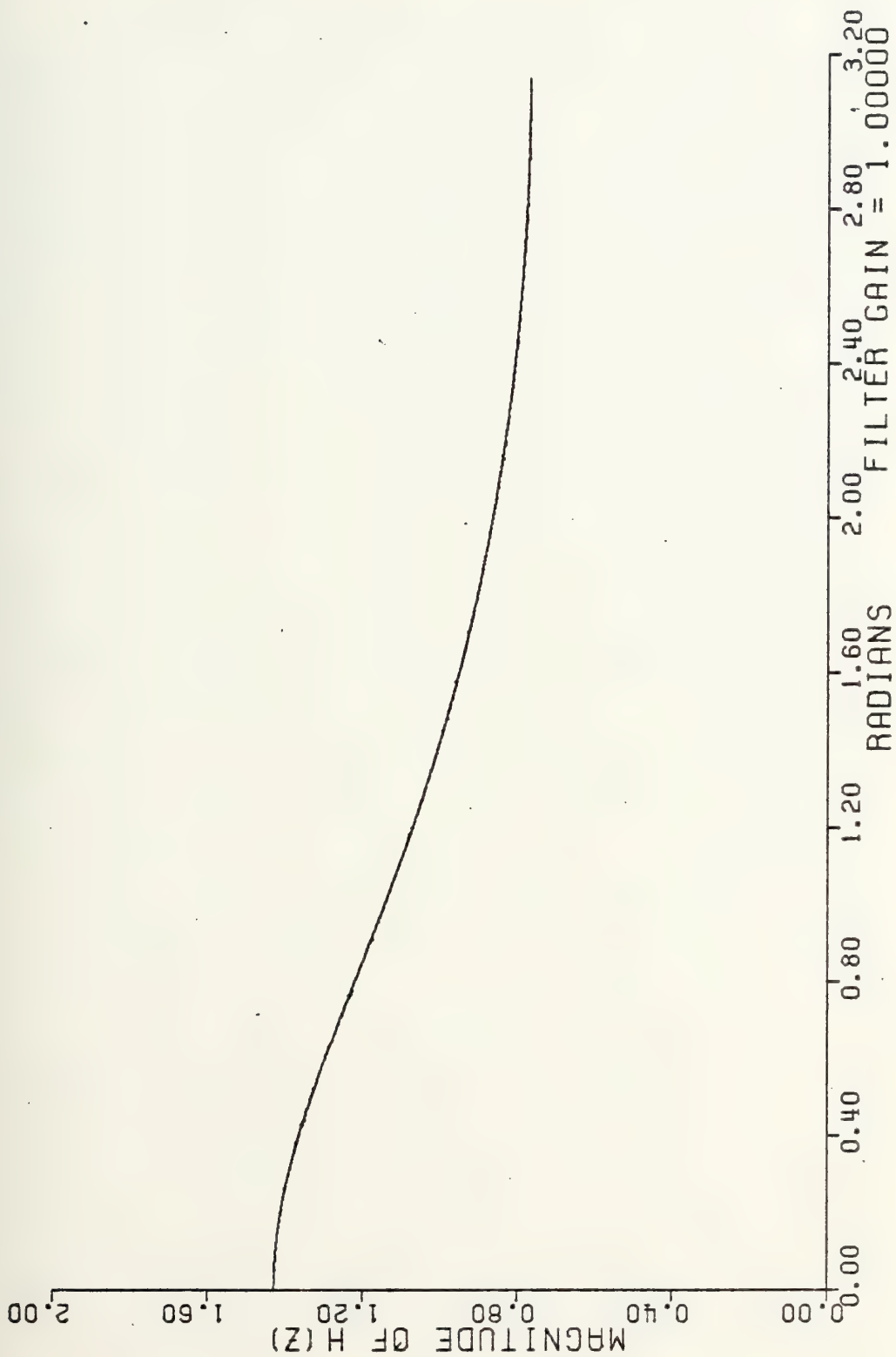


Figure 7-8e Magnitude of  $H(z)$  For A Single Real Pole At (0.3)



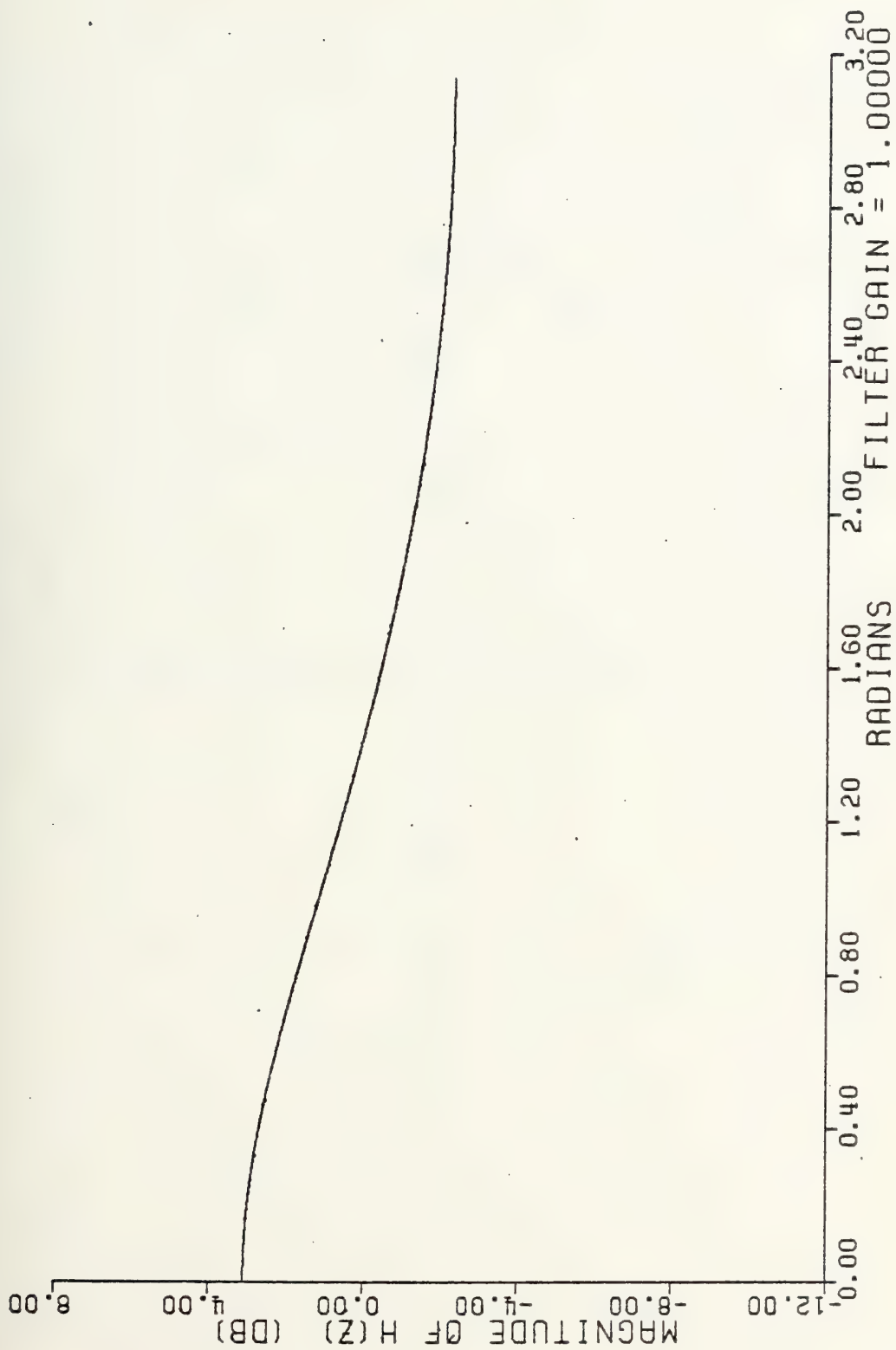


Figure 7-8f Magnitude Of  $H(z)$  In Decibels For A Single Real Pole At (0.3)



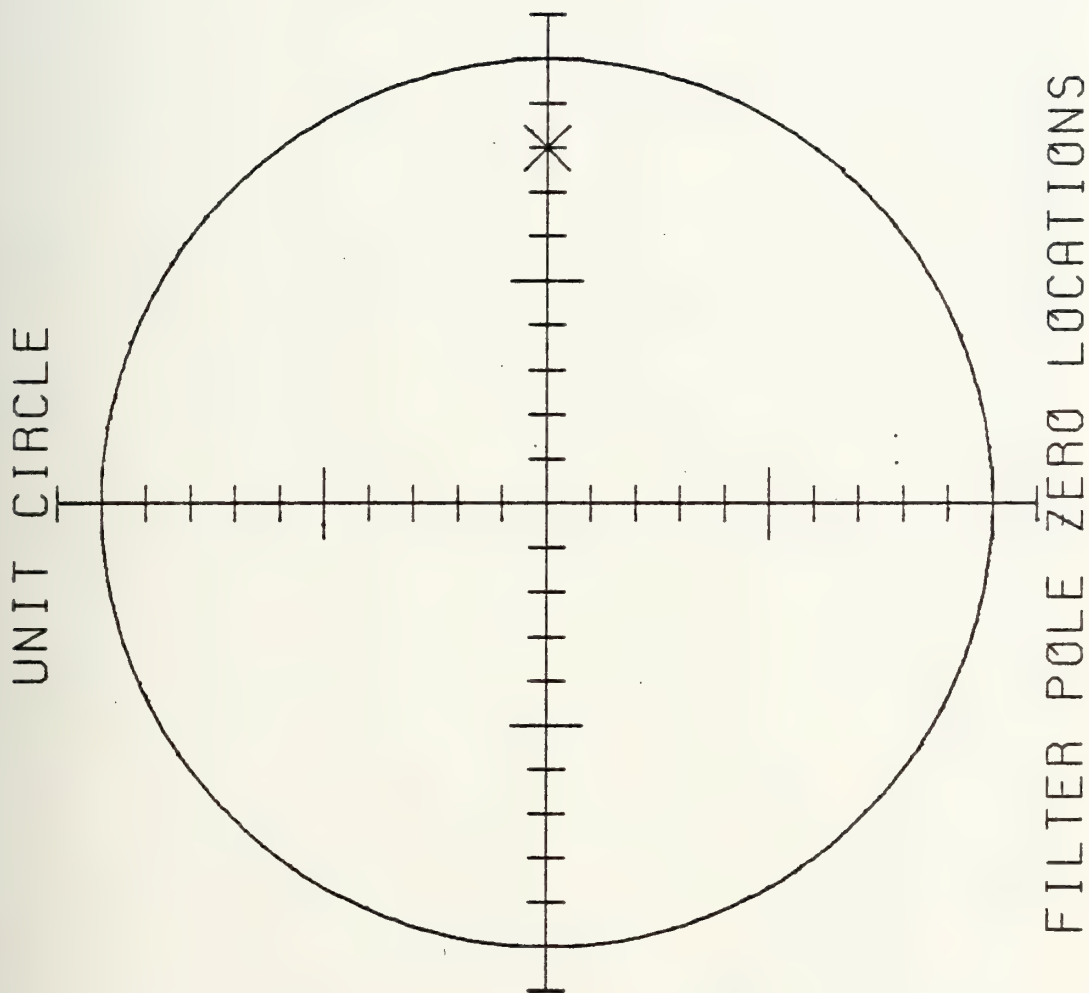


Figure 7-9a Example Of A Single Real Pole At (0.8)





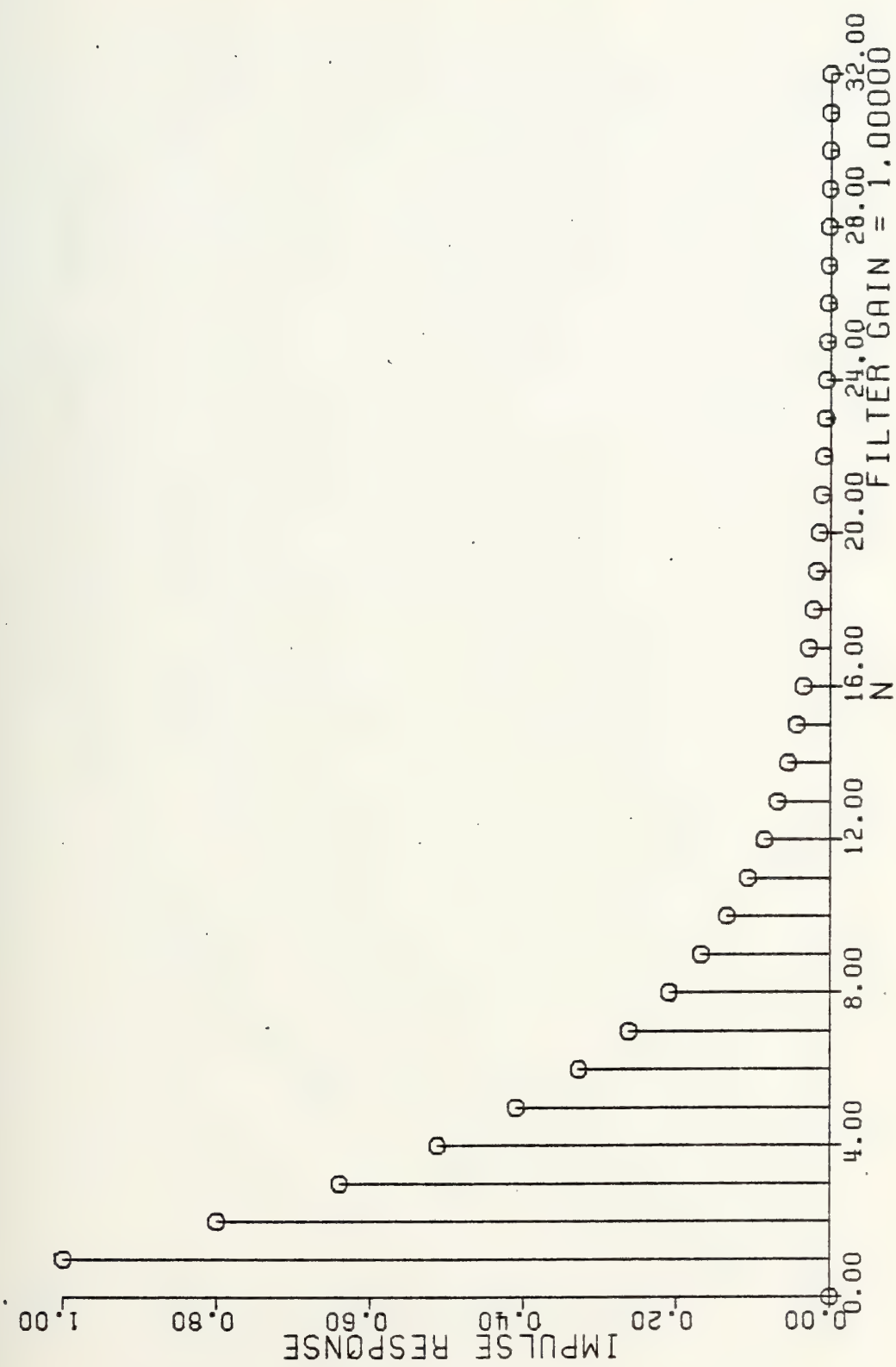


Figure 7-9b Unit Sample Response For A Single Real Pole At (0.8)



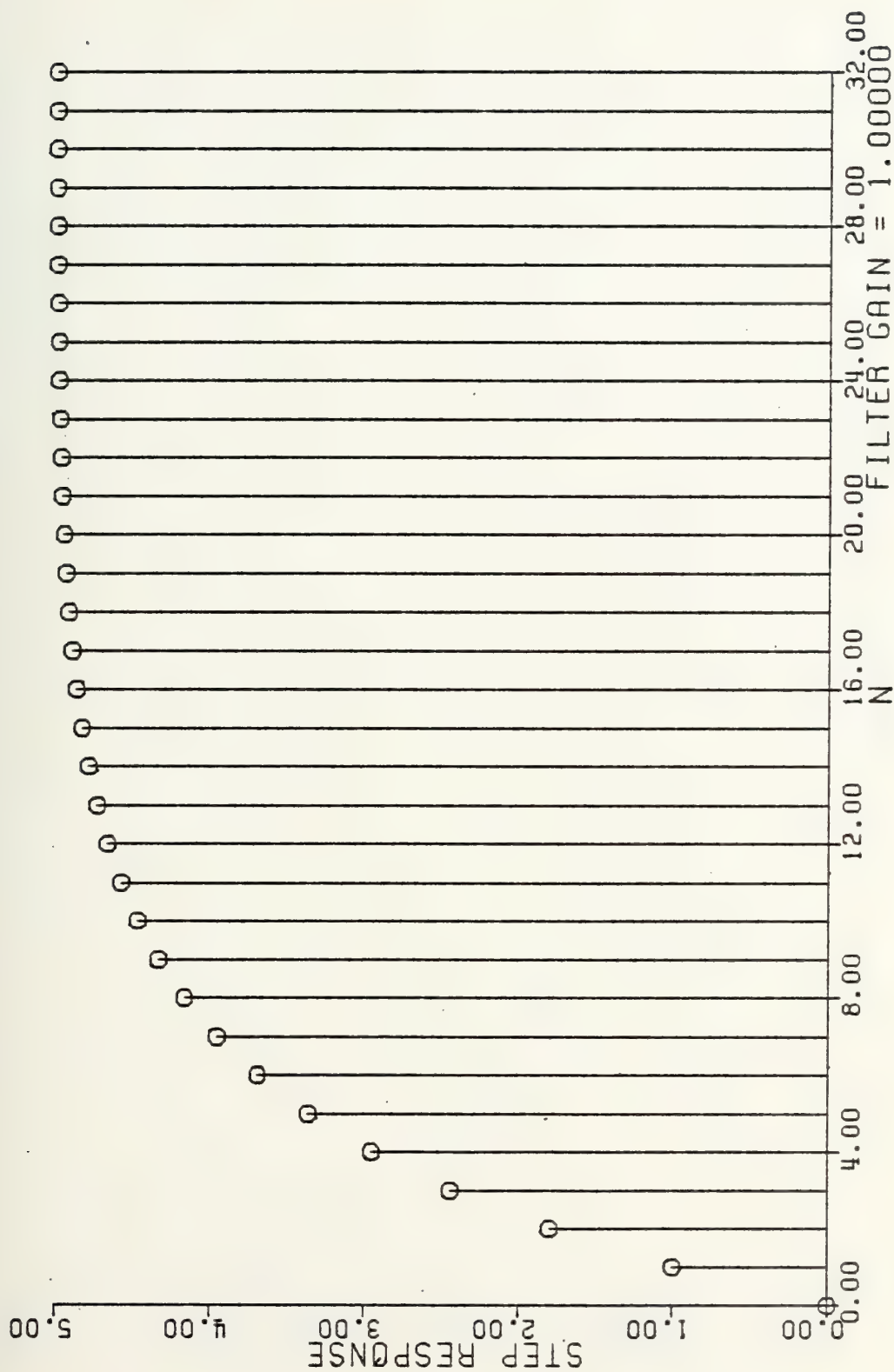


Figure 7-9c Unit Step Response For A Single Real Pole At (0.8)



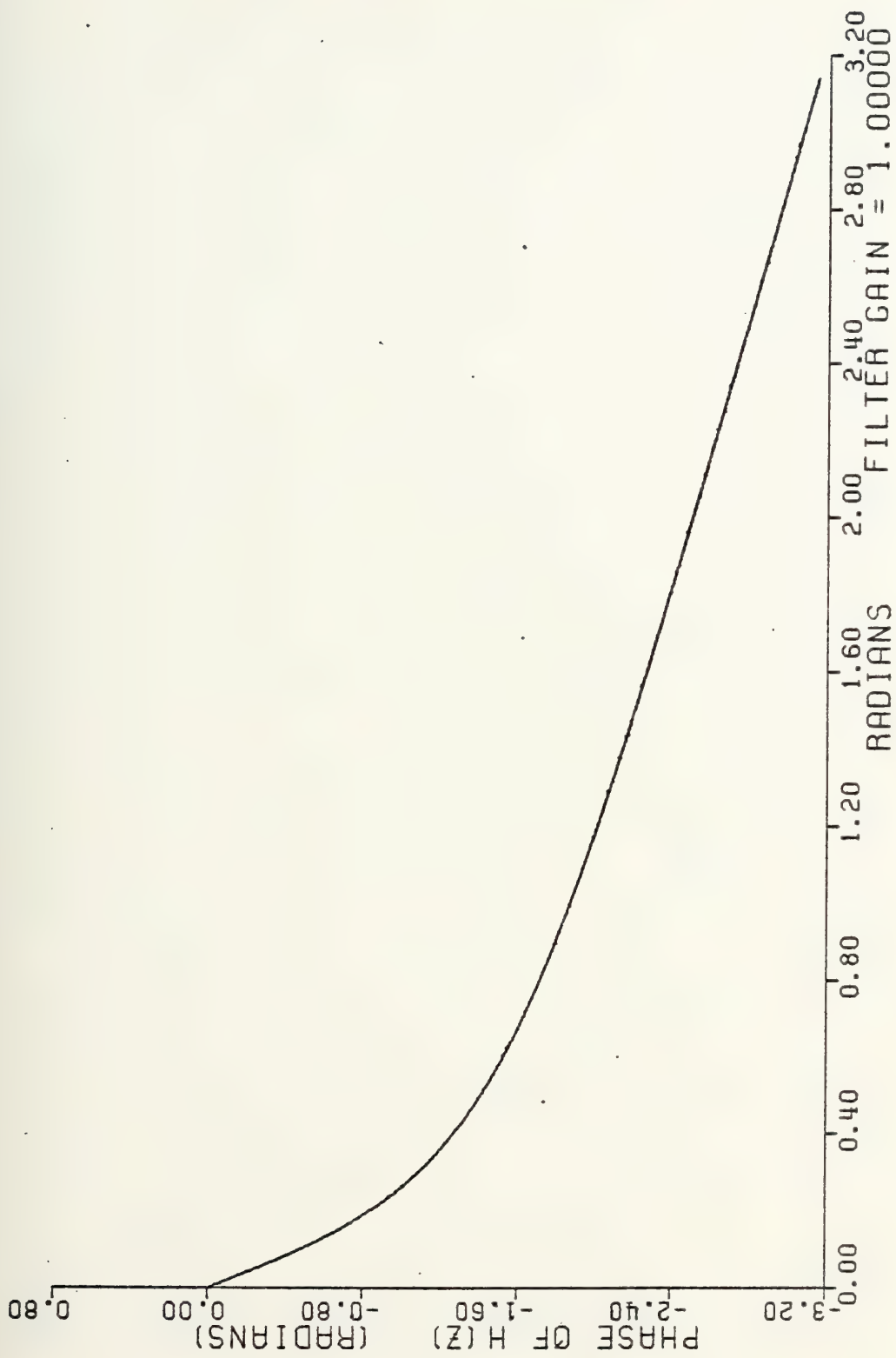


Figure 7-9d Phase Response For A Single Real Pole At (0.8)



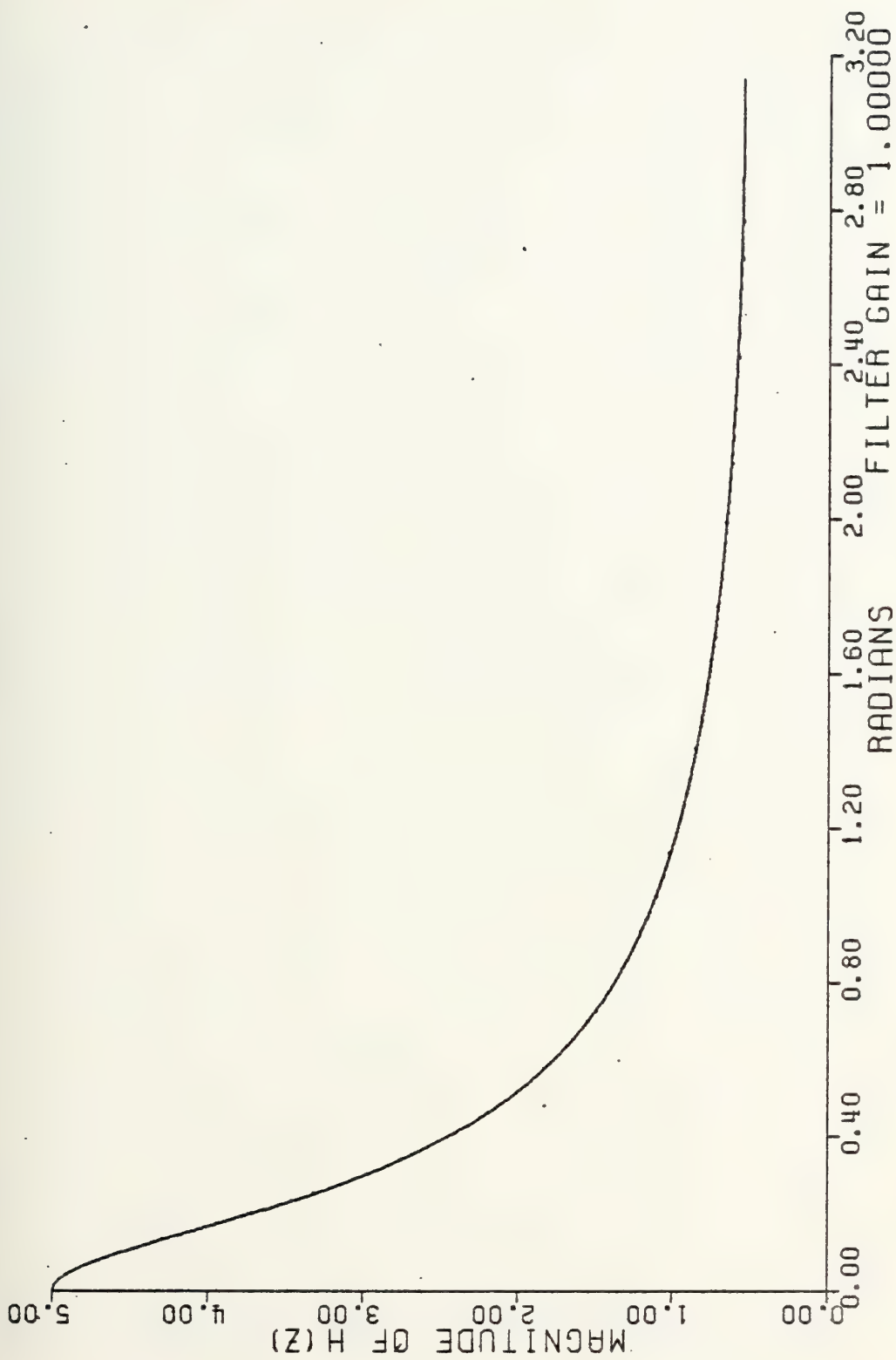


Figure 7-9e Magnitude of  $H(z)$  For A Single Real Pole At (0.8)





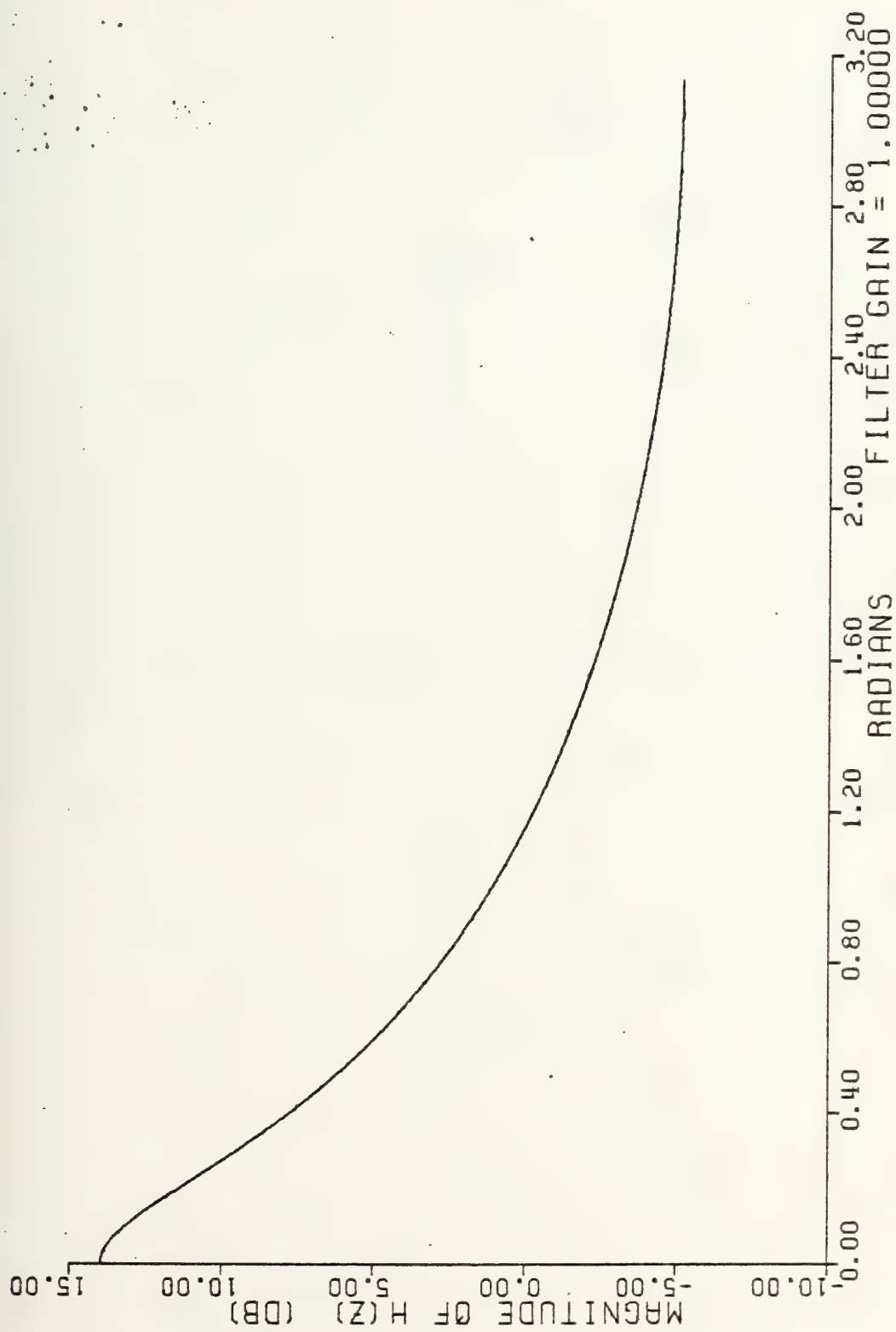


Figure 7-9f Magnitude Of  $H(z)$  For A Single Real Pole At (0.8)



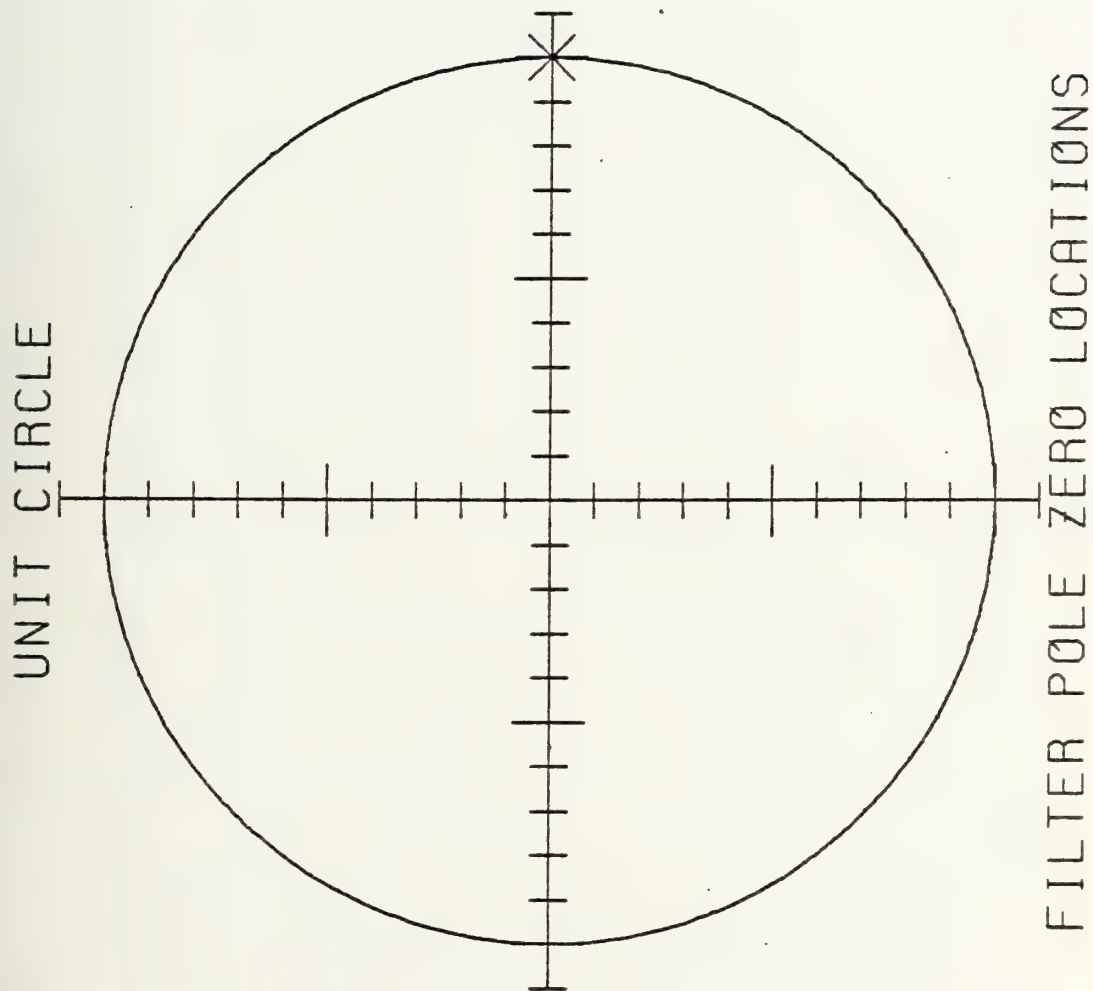


Figure 7-10a Example Of A Single Real Pole On the Unit Circle at (1.0)



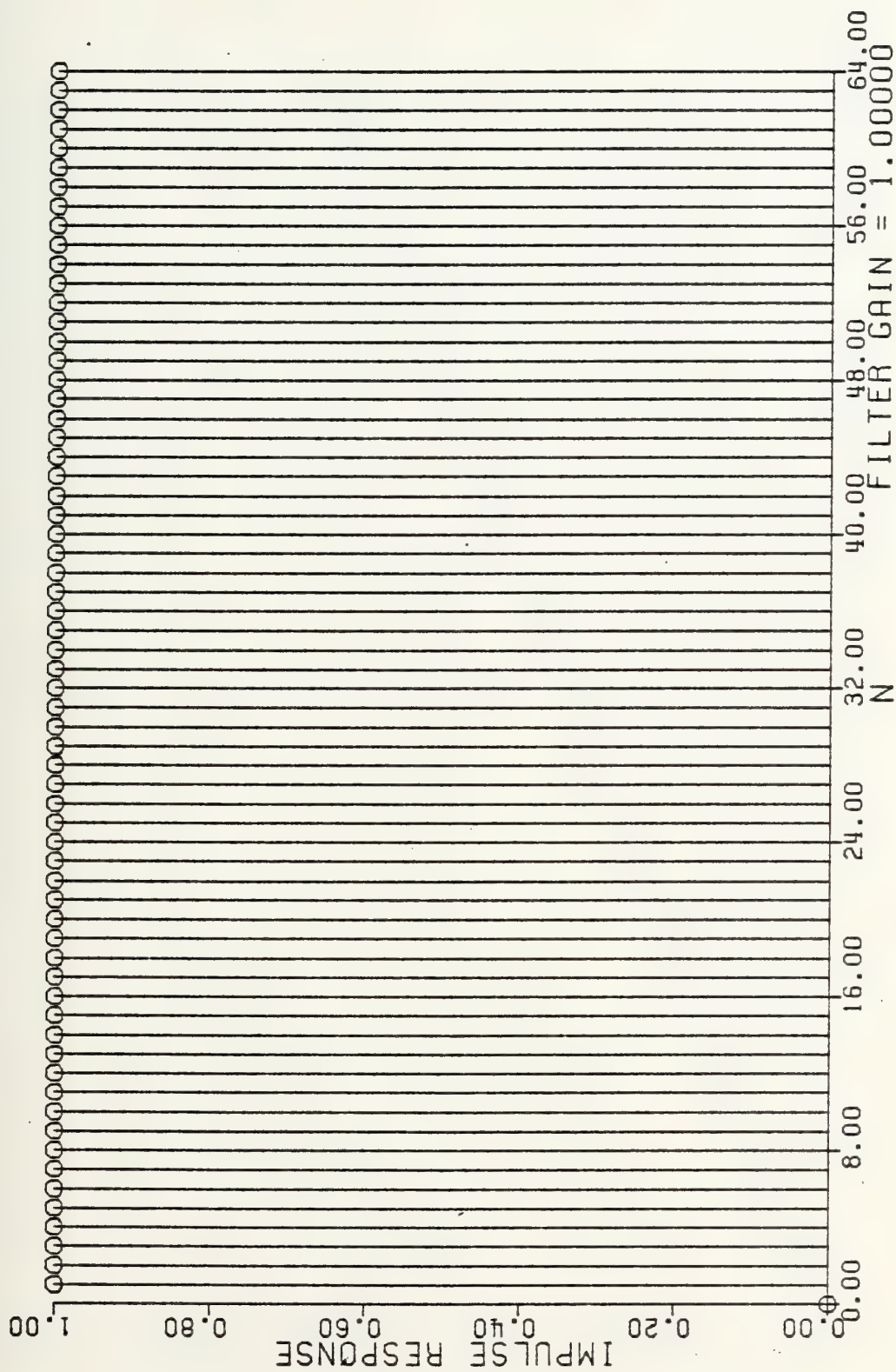


Figure 7-10b Unit Sample Response For A Single Real Pole At (1.0)



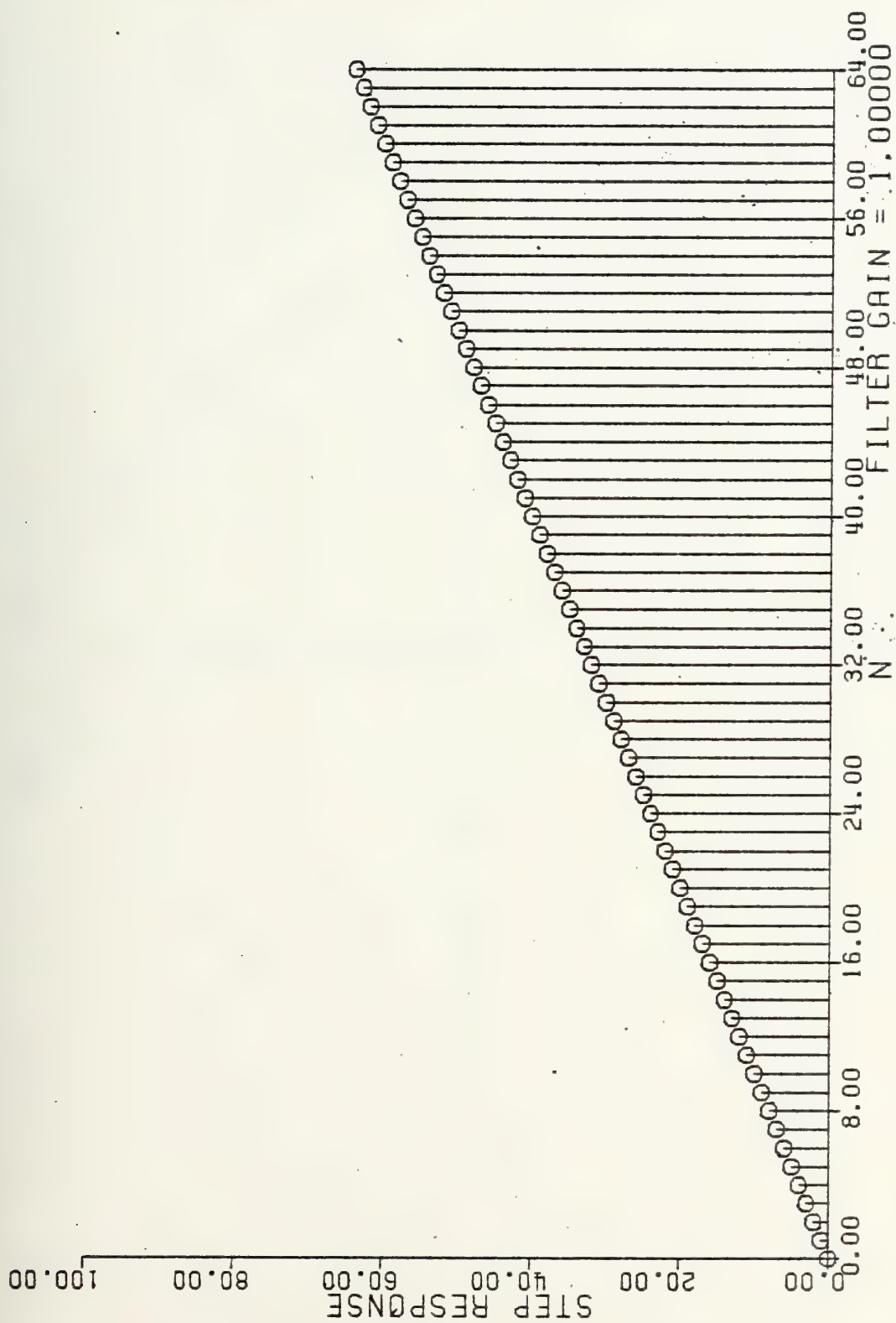


Figure 7-10c Unit Step Response For A Single Real Pole At (1.0)





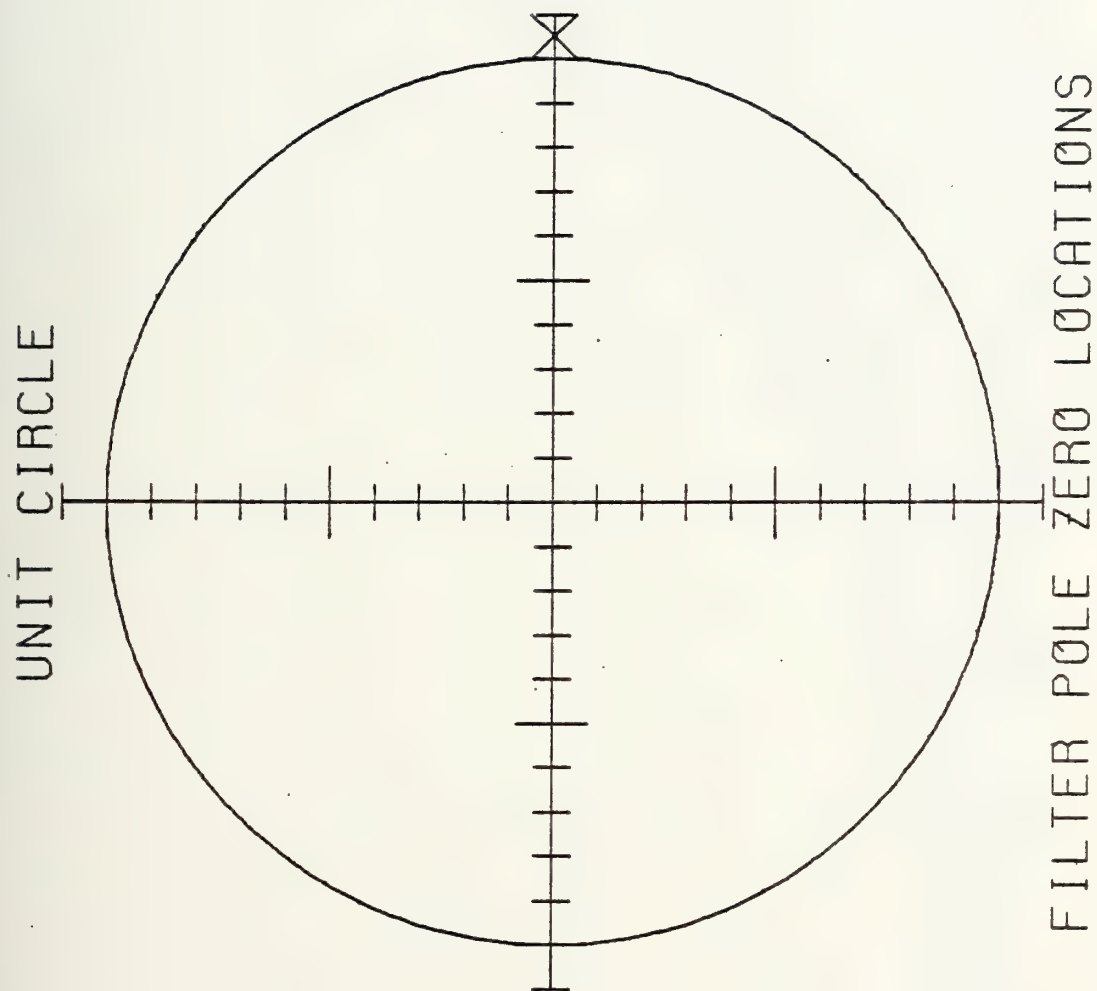


Figure 7-11a Example Of A Single Real Pole Outside The Unit Circle At (1.05)



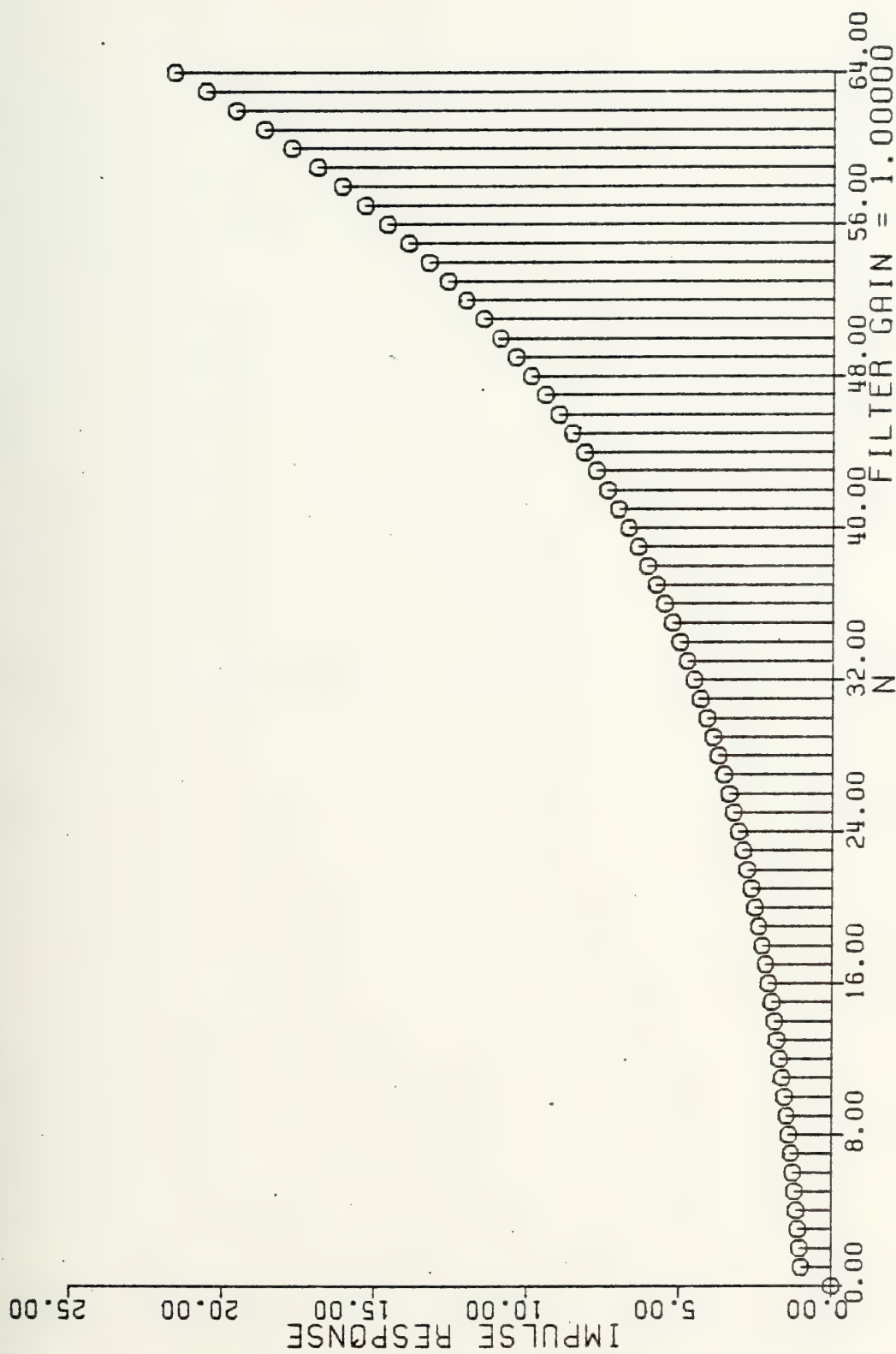


Figure 7-11b Unit Sample Response For A Single Real Pole At (1.05)



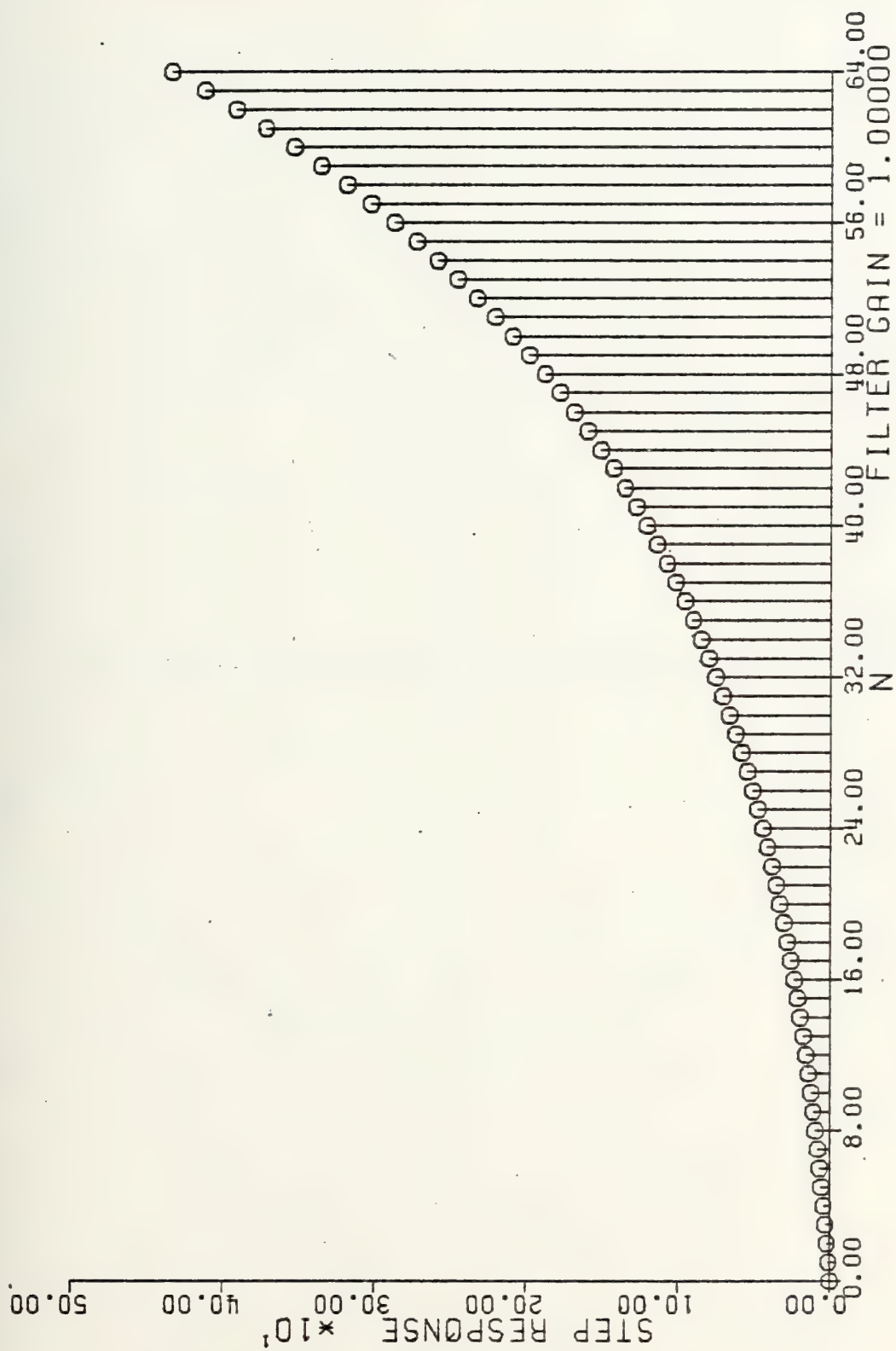


Figure 7-11c Unit Step Response For A Single Real Pole At (1.05)



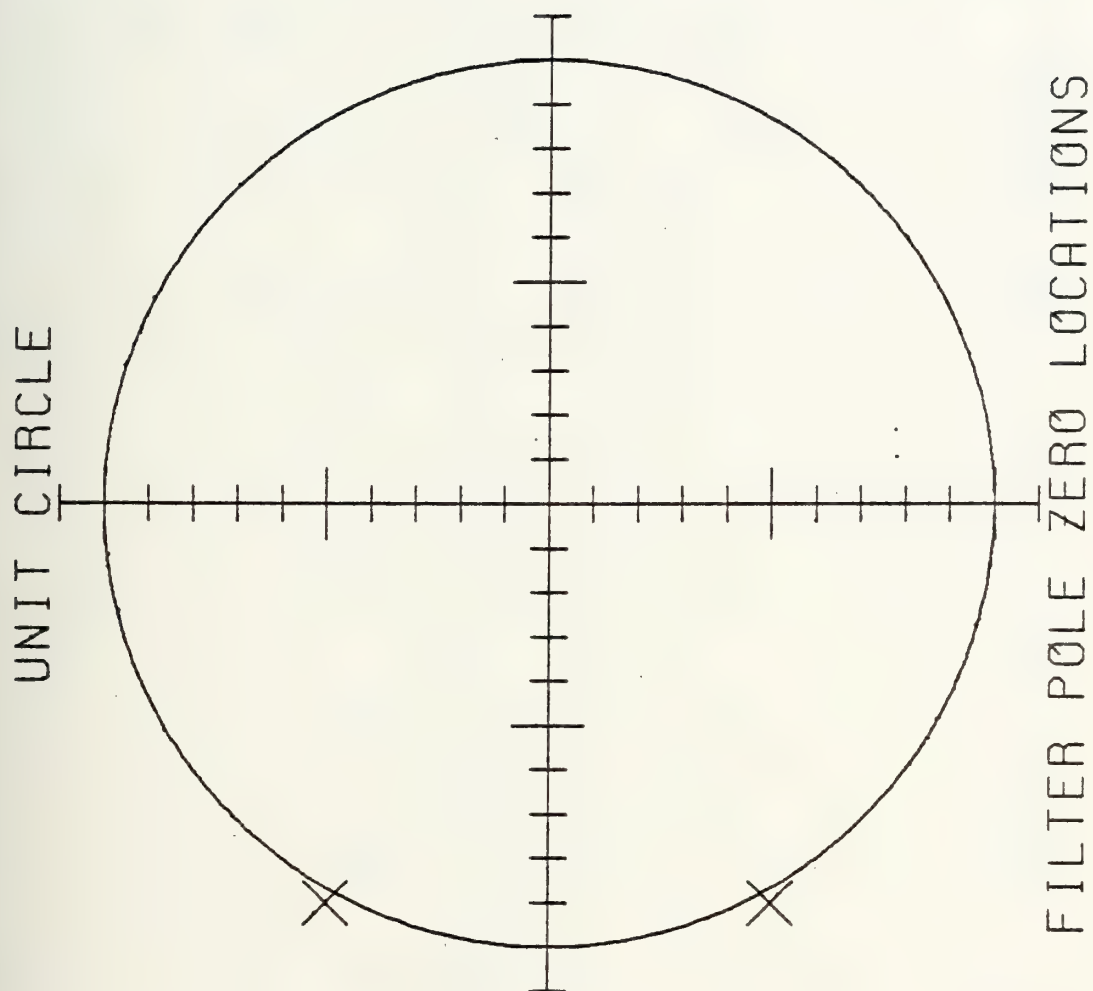


Figure 7-12a Example Of A Complex Pole Pair At  $(-0.8 \pm j0.5)$





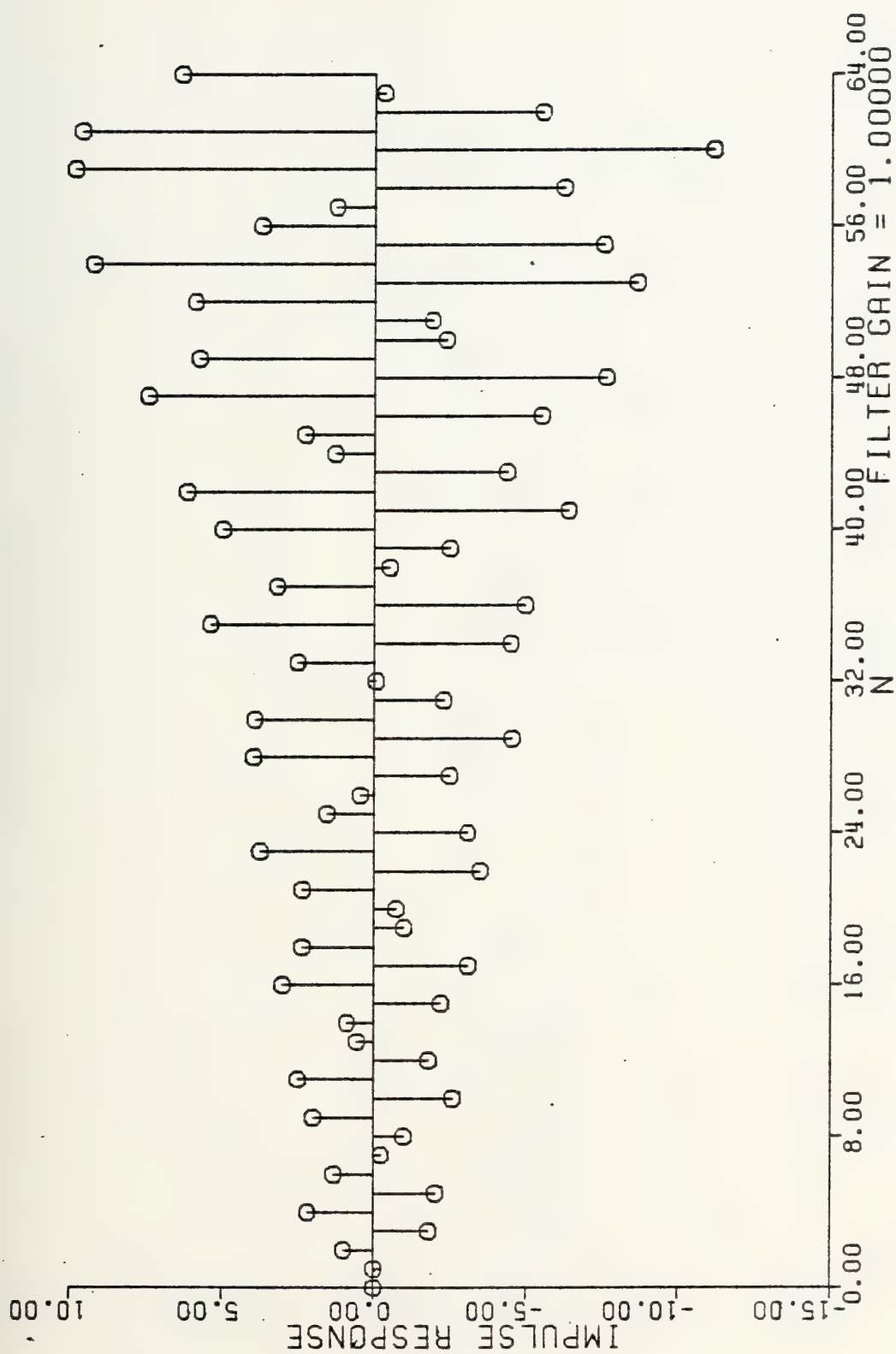


Figure 7-12b Unit Sample Response For A Complex Pole Pair At  $(-0.8 \pm j0.5)$



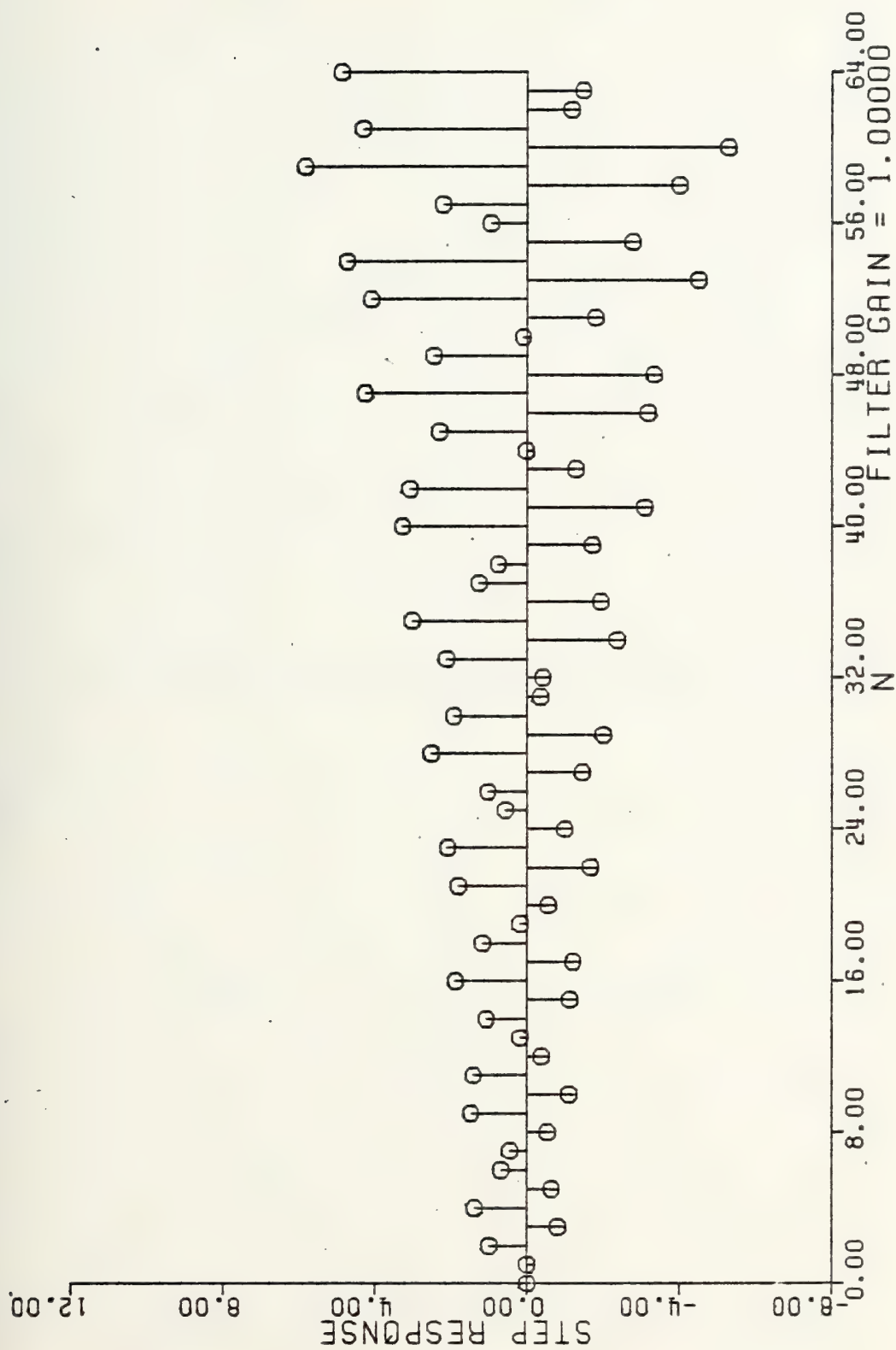


Figure 7-12c Unit Step Response For A Complex Pole Pair At  $(0.8 + j0.5)$



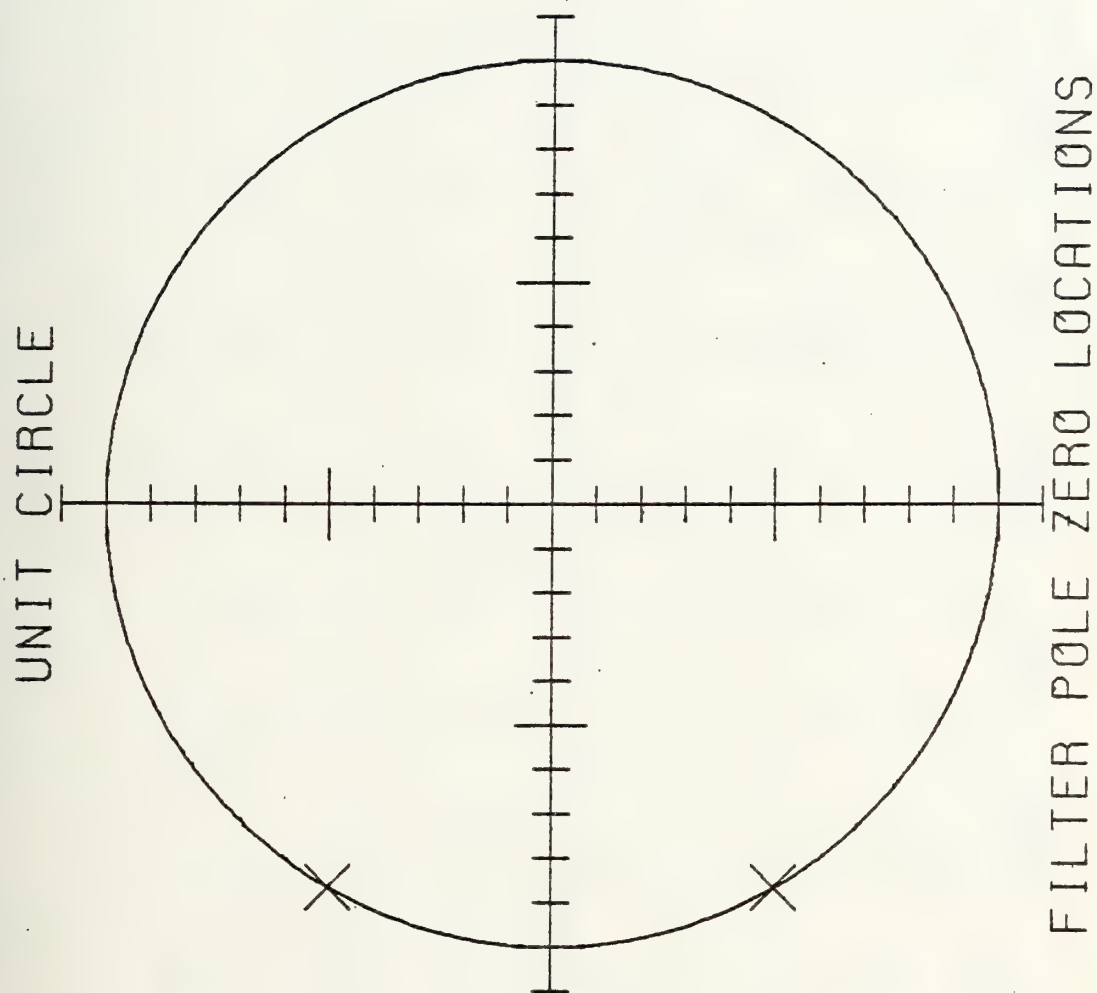


Figure 7-13a Example Of A Complex Pole Pair On The Unit Circle At  $(-.866025 + j0.5)$



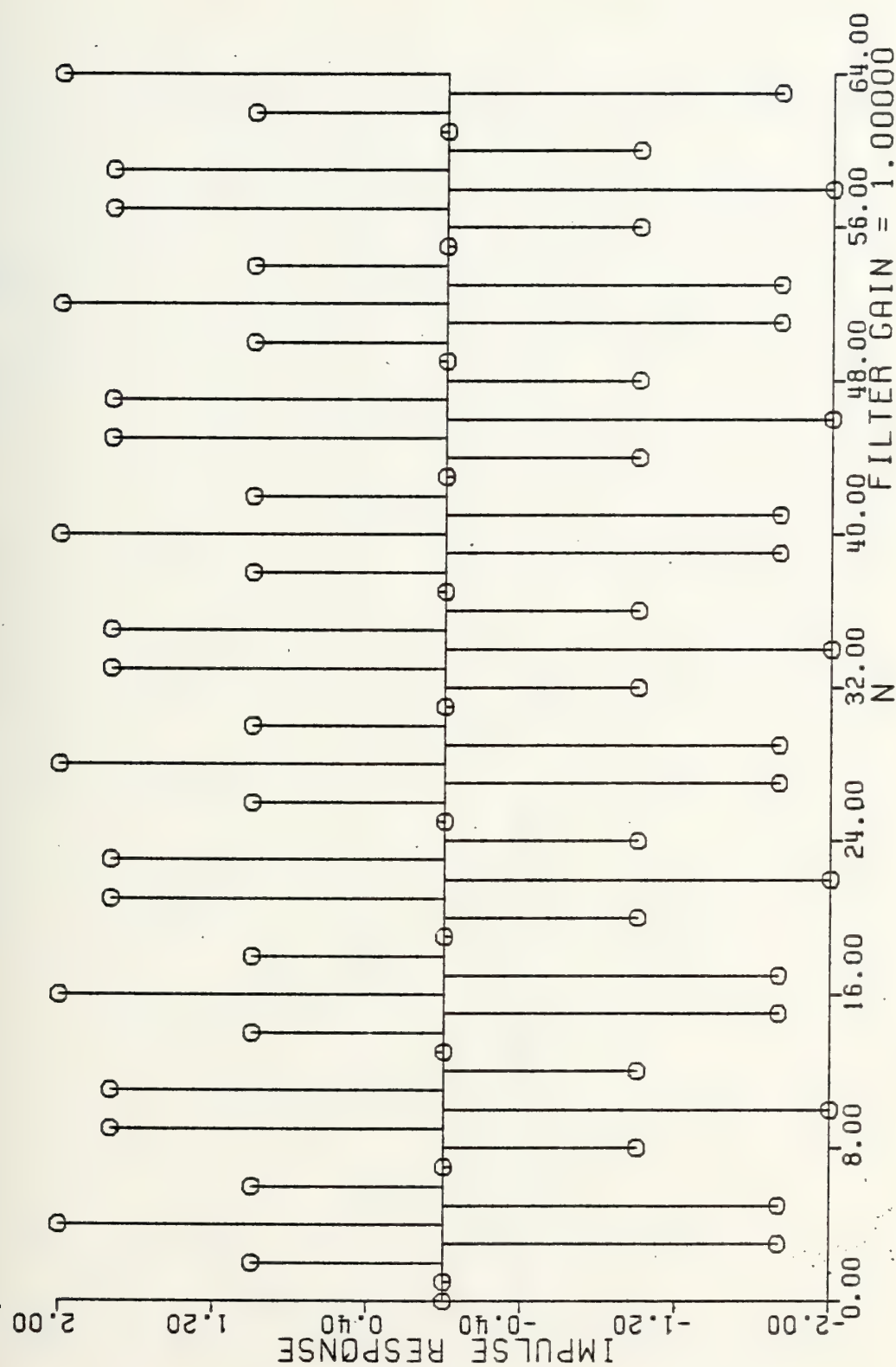


Figure 7-13b Unit Sample Response Of A Complex Pole Pair At  $(-.866025 + j0.5)$





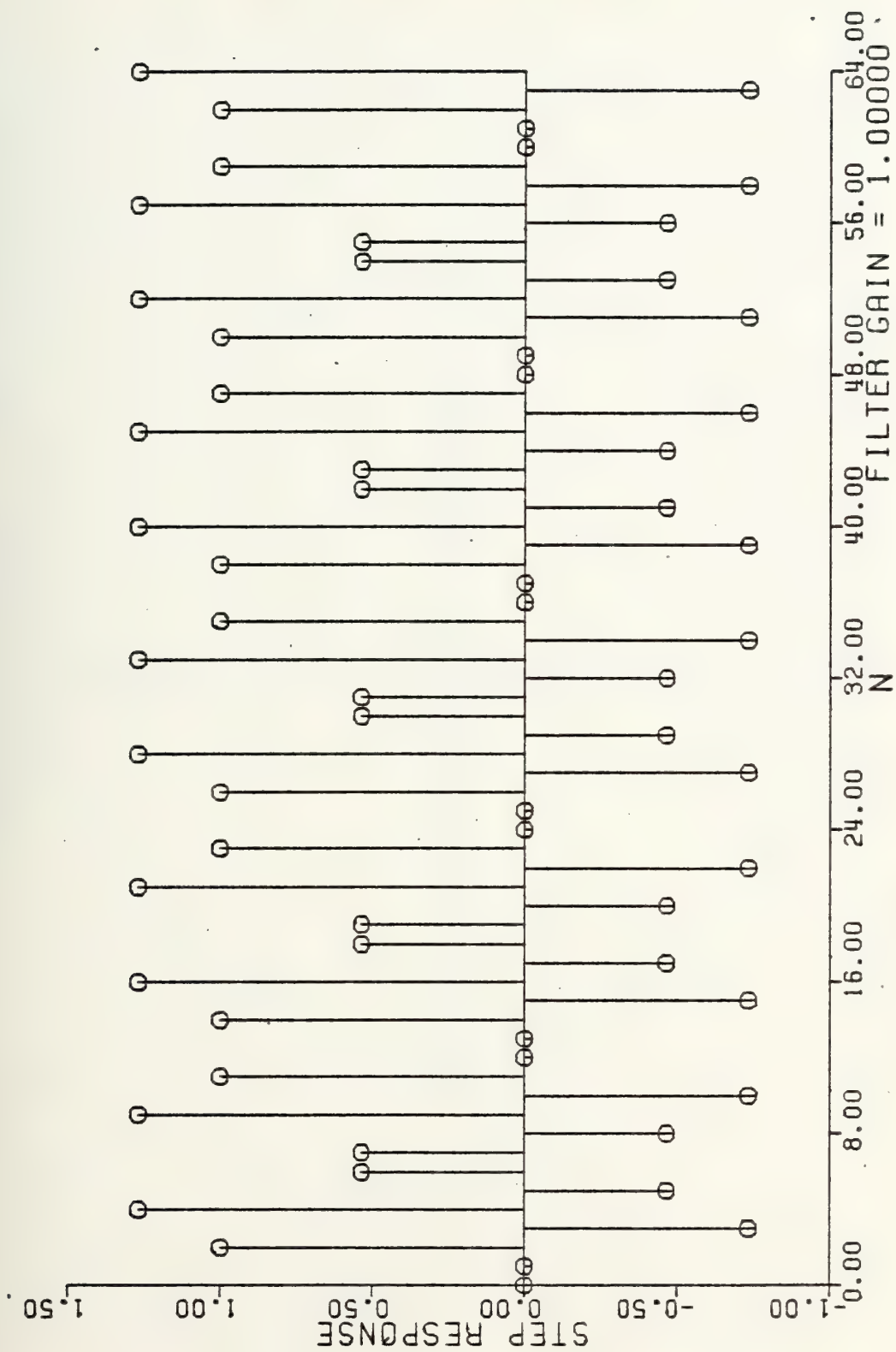


Figure 7-13c Unit Step Response Of A Complex Pole Pair At  $(-0.866025 + j0.5)$



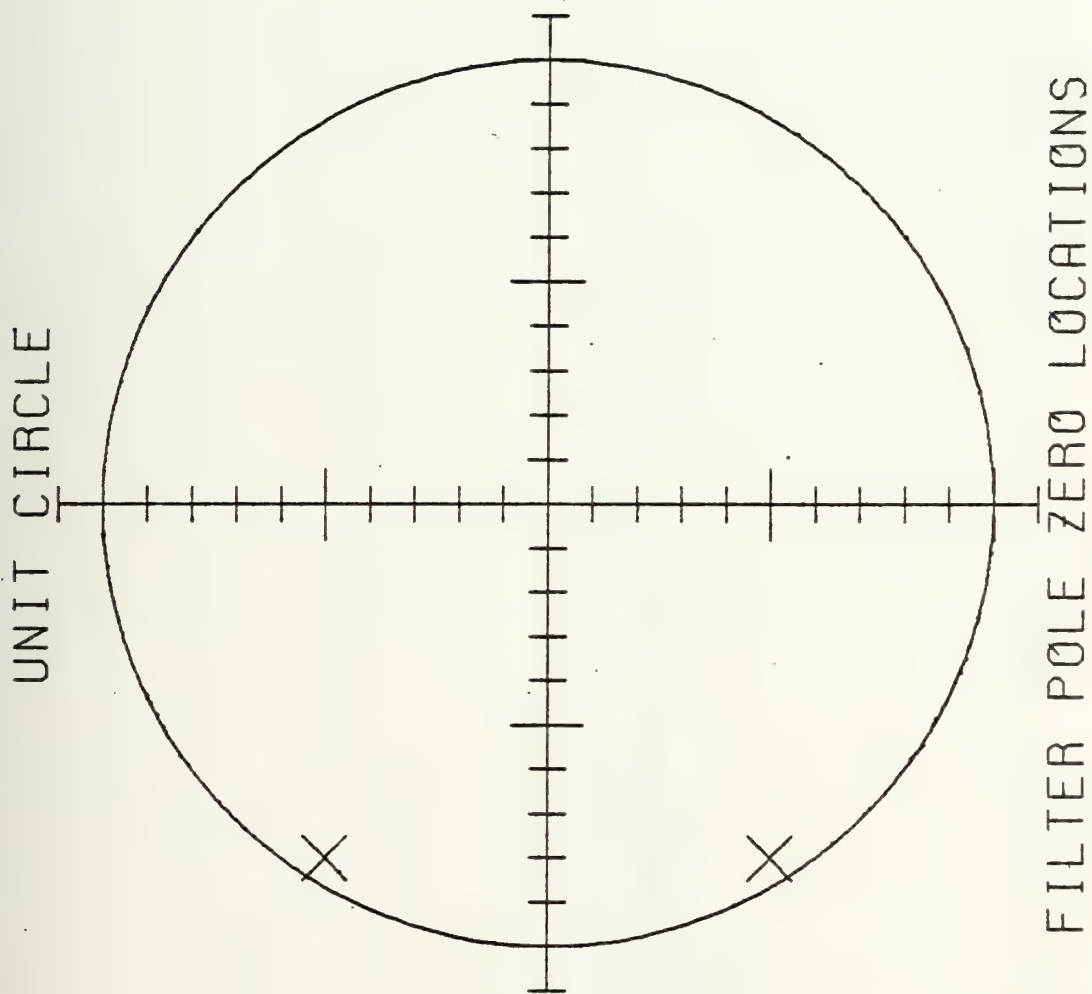


Figure 7-14a Example Of A Complex Pole Pair At  $(-0.8 \pm j0.5)$



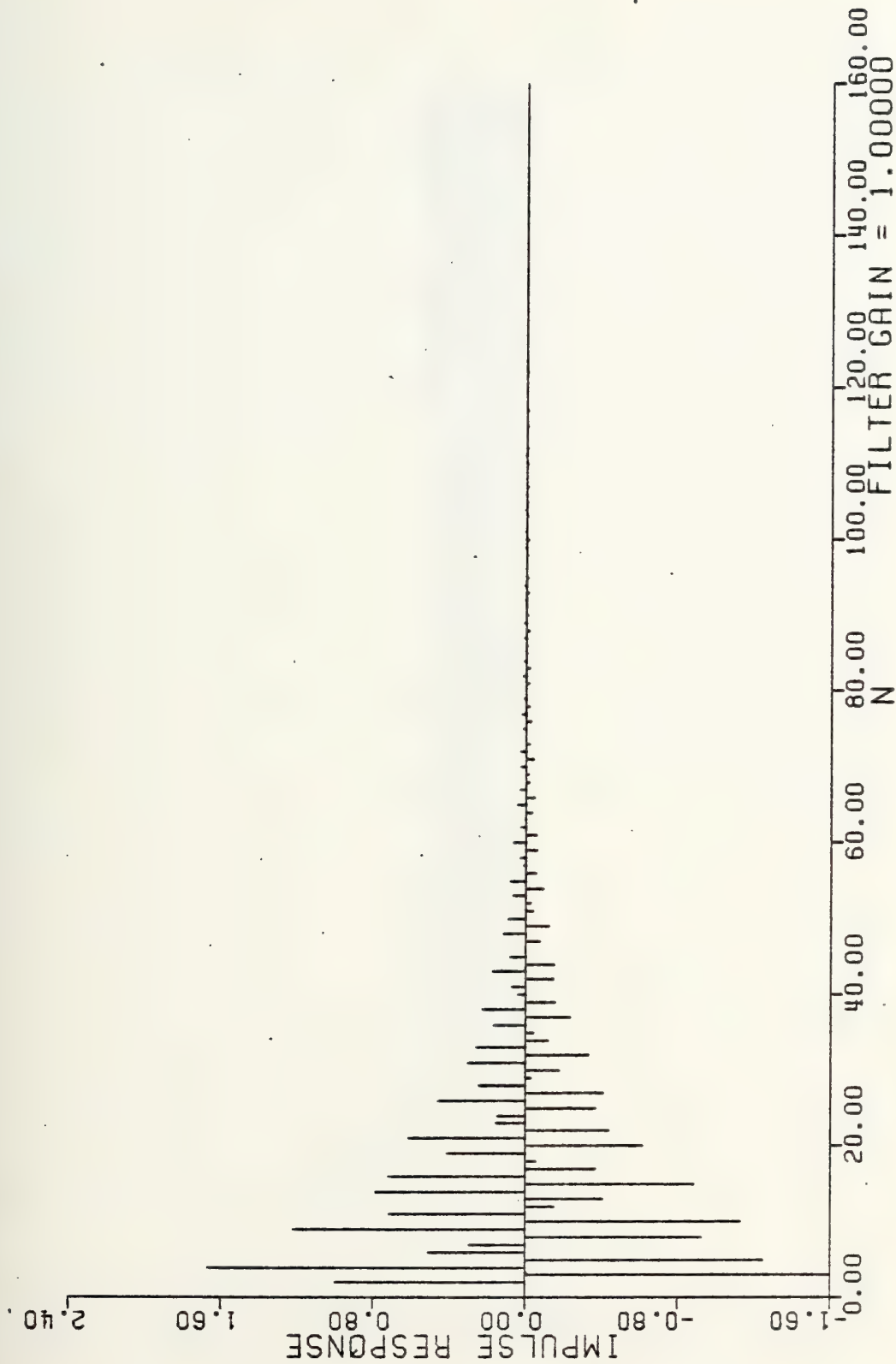


Figure 7-14b Unit Sample Response For A Complex Pole Pair At  $(-0.8 + j0.5)$



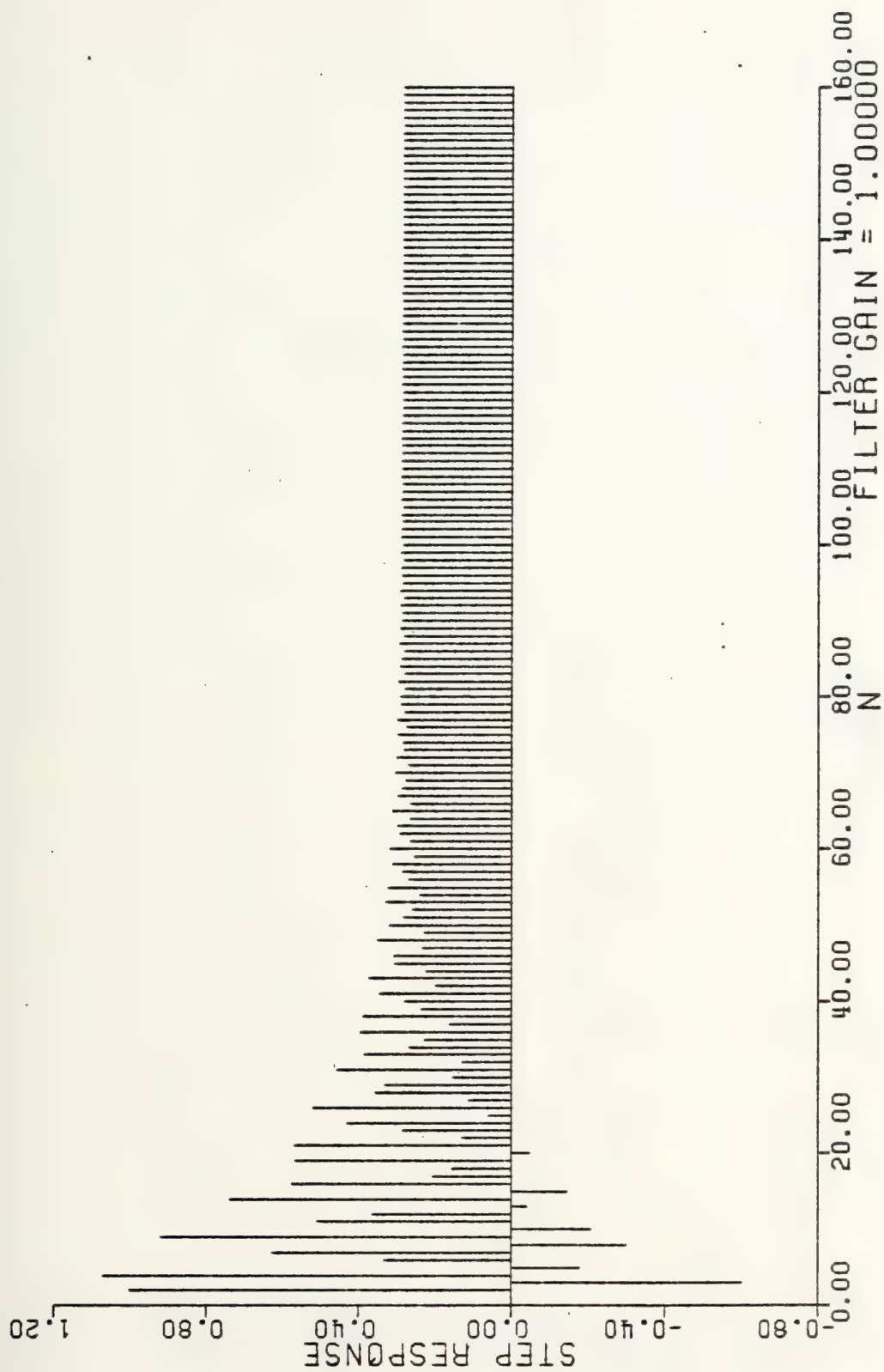


Figure 7.14c Unit Step Response For a Complex Pole Pair At  $(-0.8 + j0.5)$





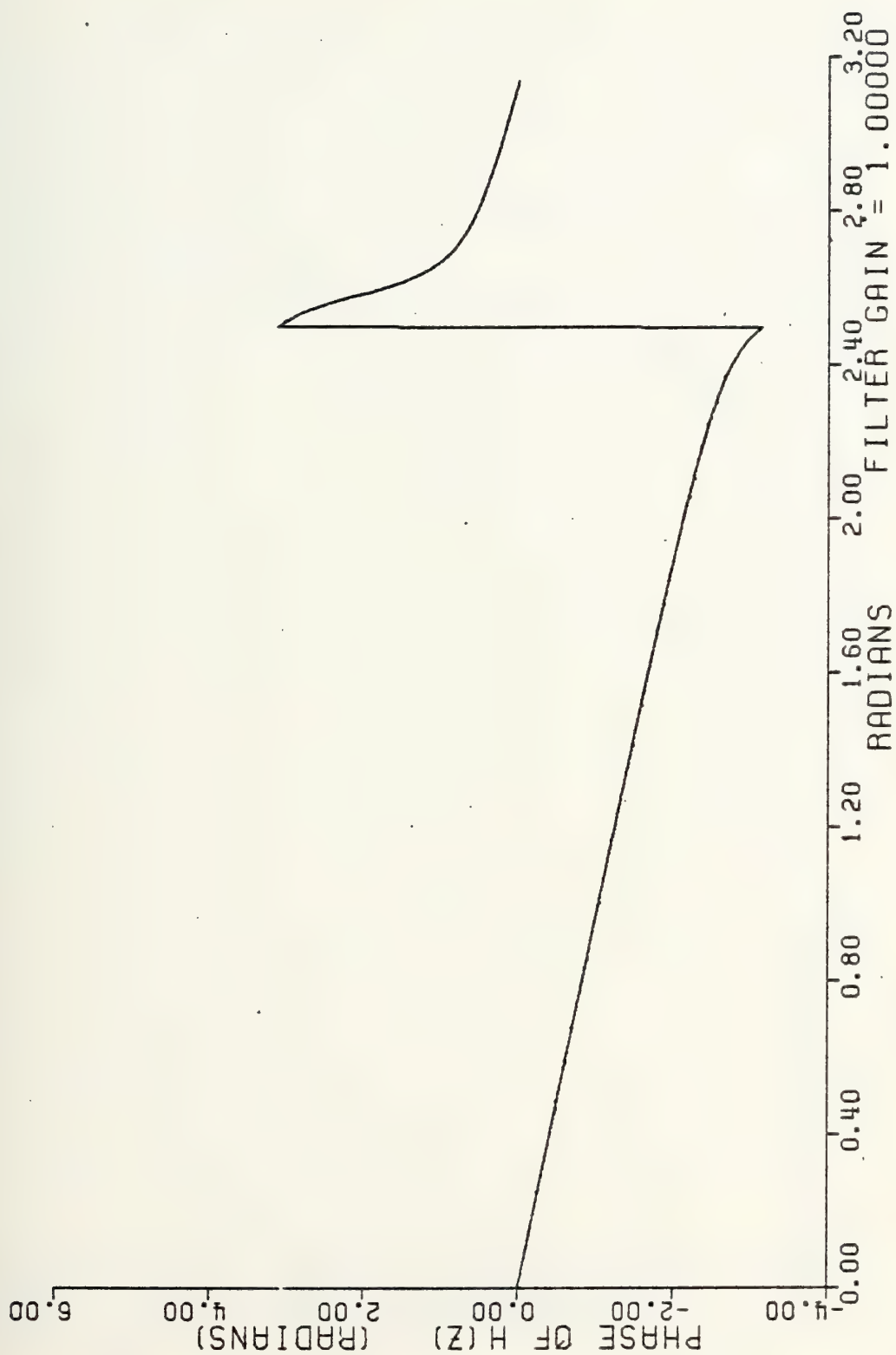


Figure 7-14d Phase Response For A Complex Pole Pair At  $(-0.8 \pm j0.5)$



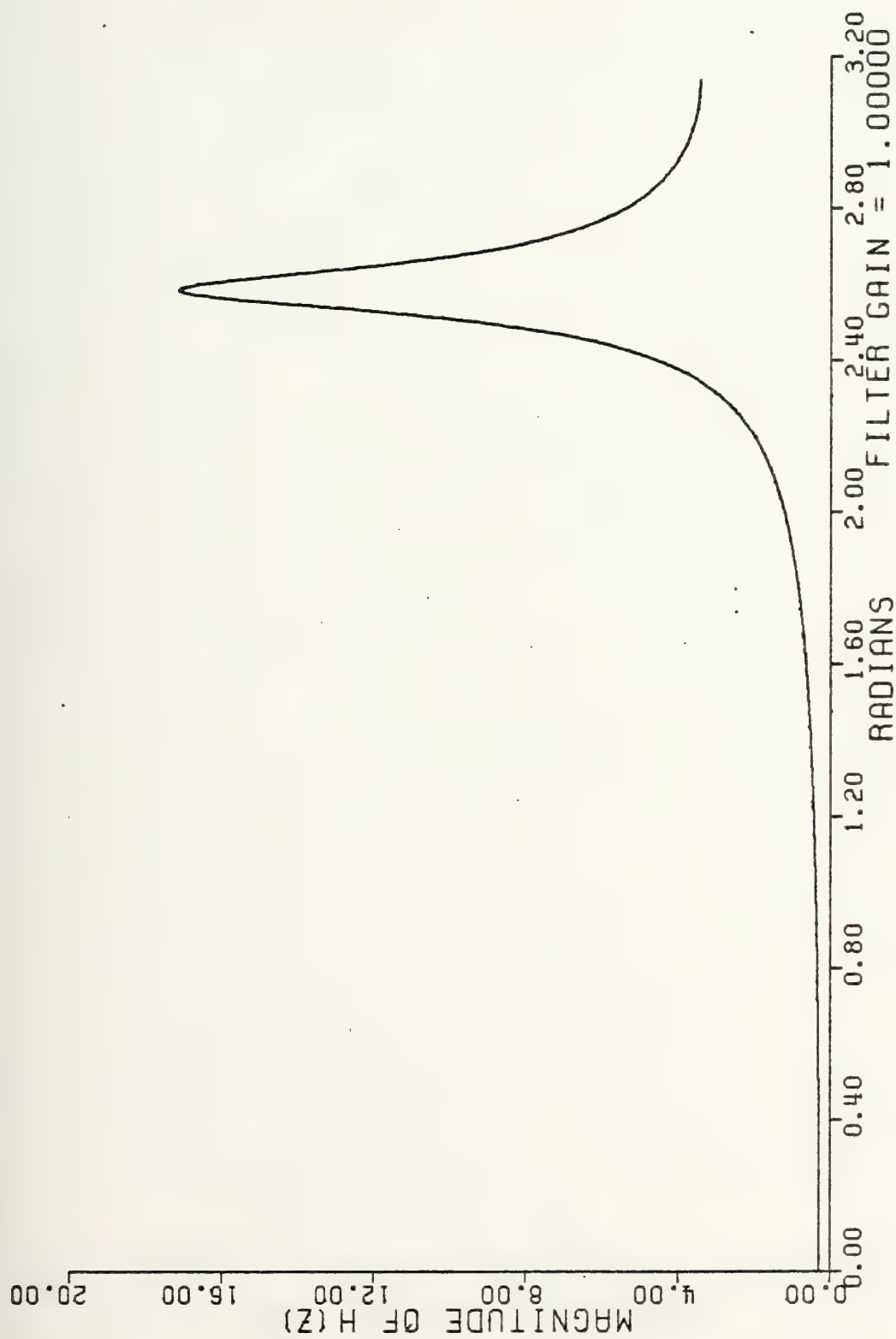


Figure 7-14e Magnitude Of  $H(z)$  For A Complex Pole Pair At  $(-0.8 + j0.5)$



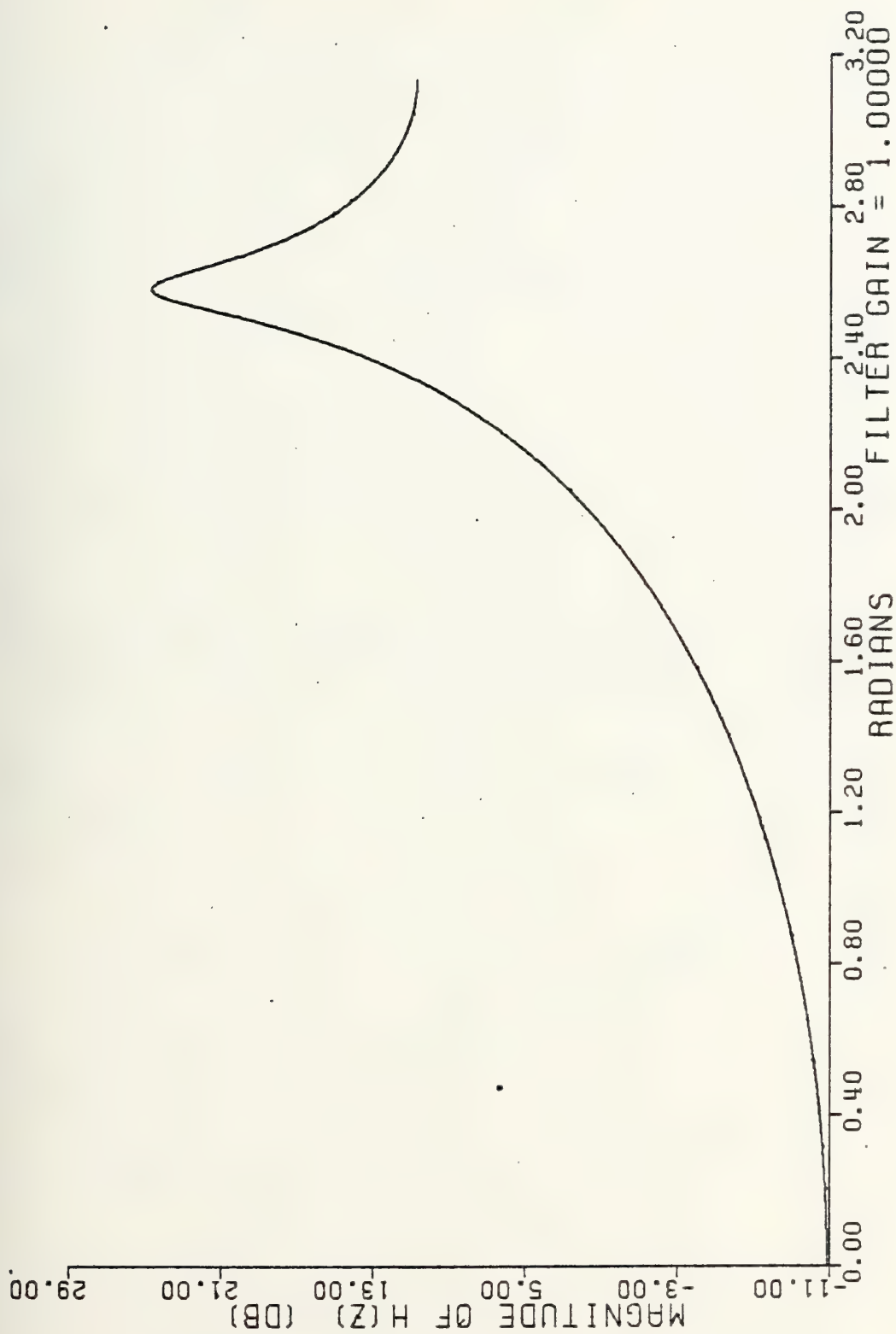


Figure 7-14f Magnitude Of  $H(z)$  In Decibels For A Complex Pole Pair At  $(-0.8 + j0.5)$



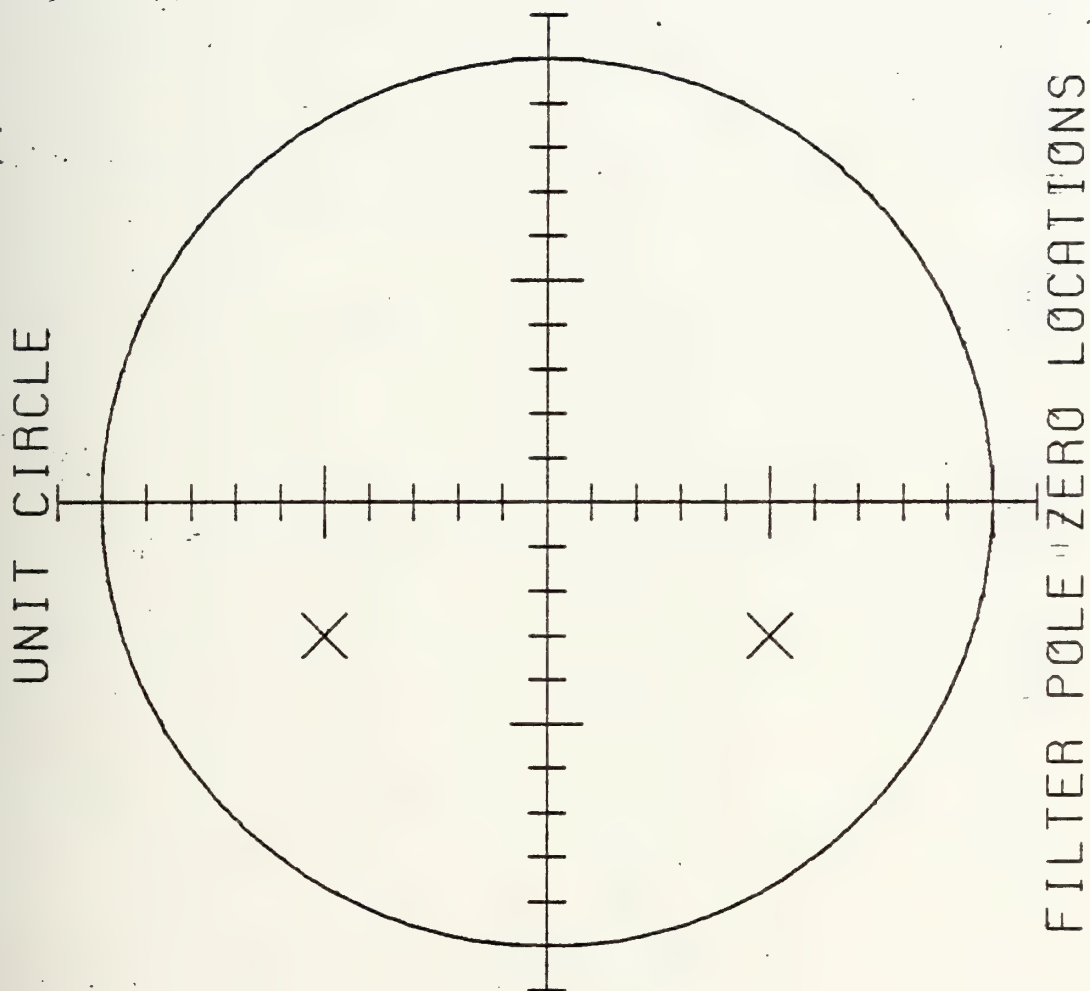


Figure 7-15a Example Of A Complex Pole Pair At  $(-0.3 \pm j0.5)$





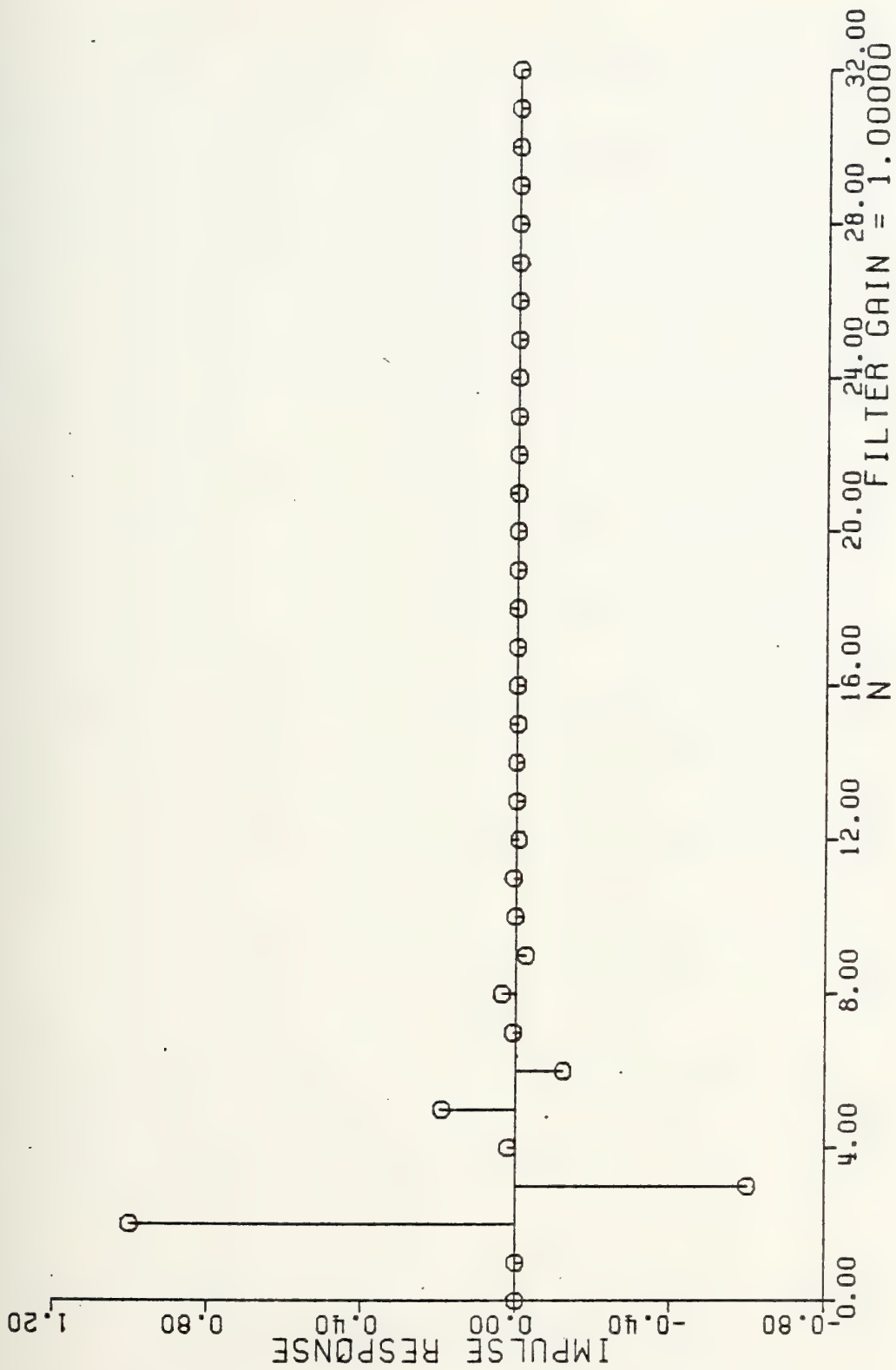


Figure 7-15b Unit Sample Response For A Complex Pole Pair At  $(-0.3 + j0.5)$



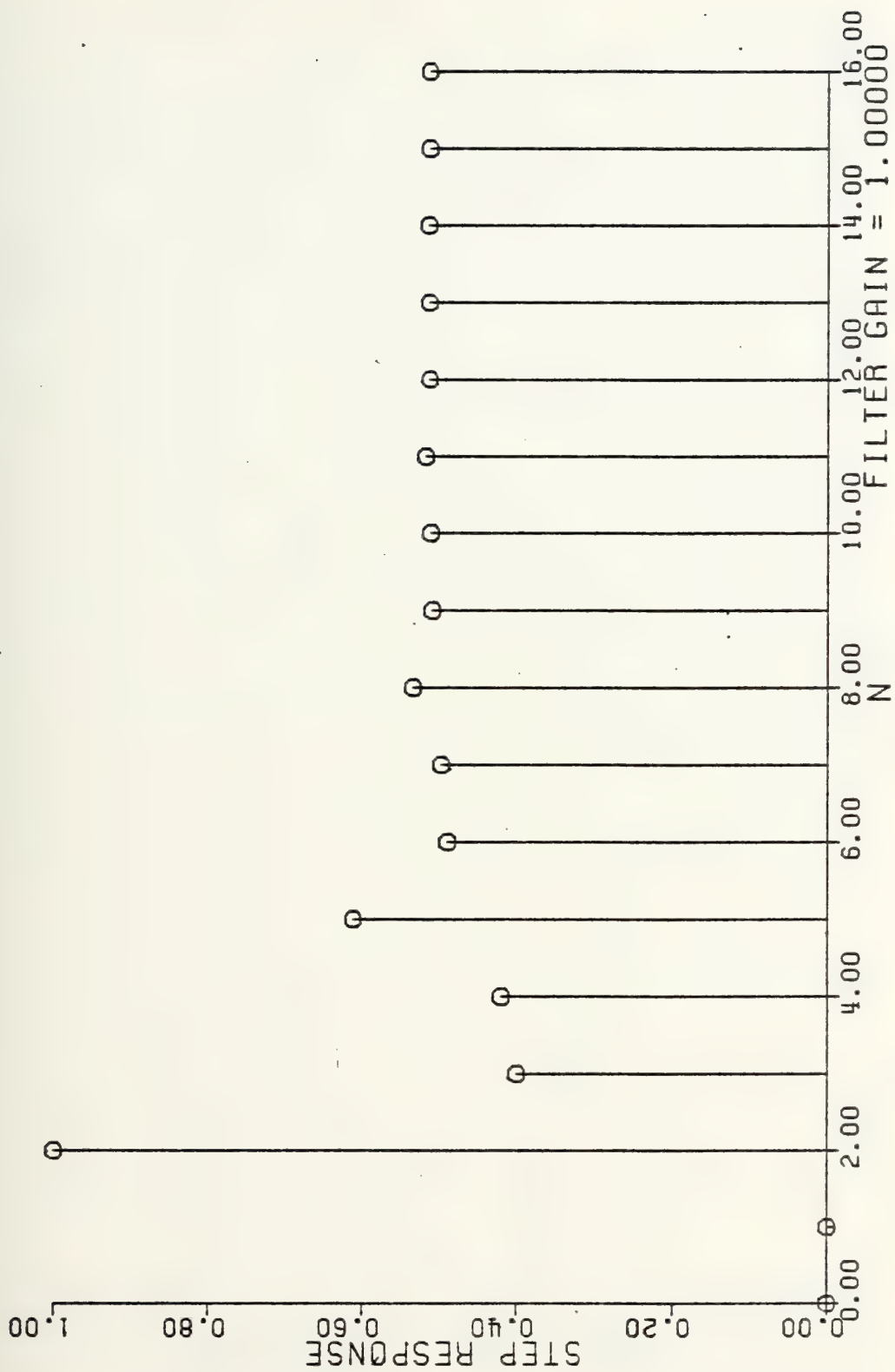


Figure 7-15c Unit Step Response For A Complex Pole Pair At  $(-0.3 + j0.5)$



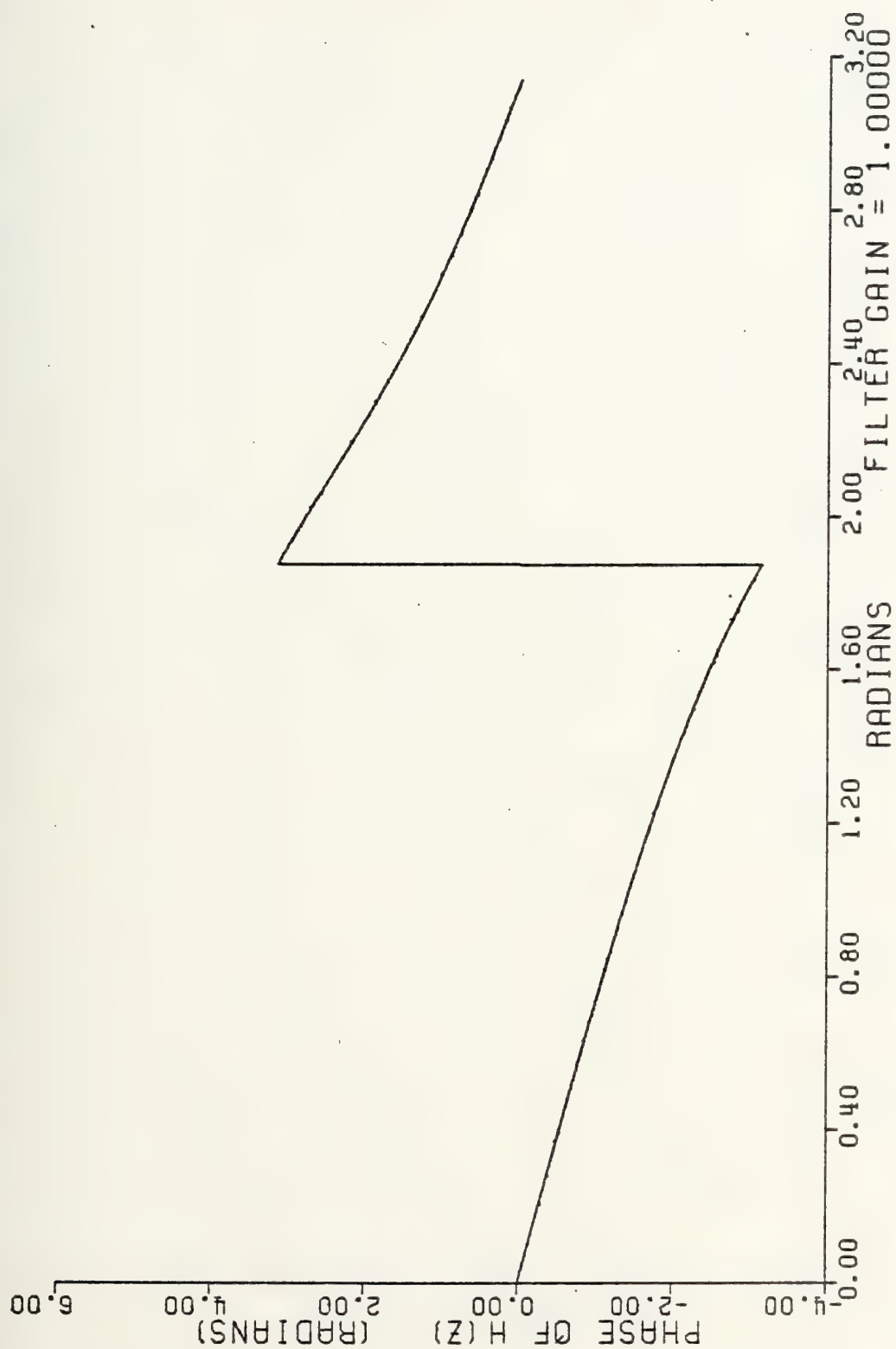


Figure 7-15d Phase Response For A Complex Pole Pair At  $(-0.3 + j0.5)$



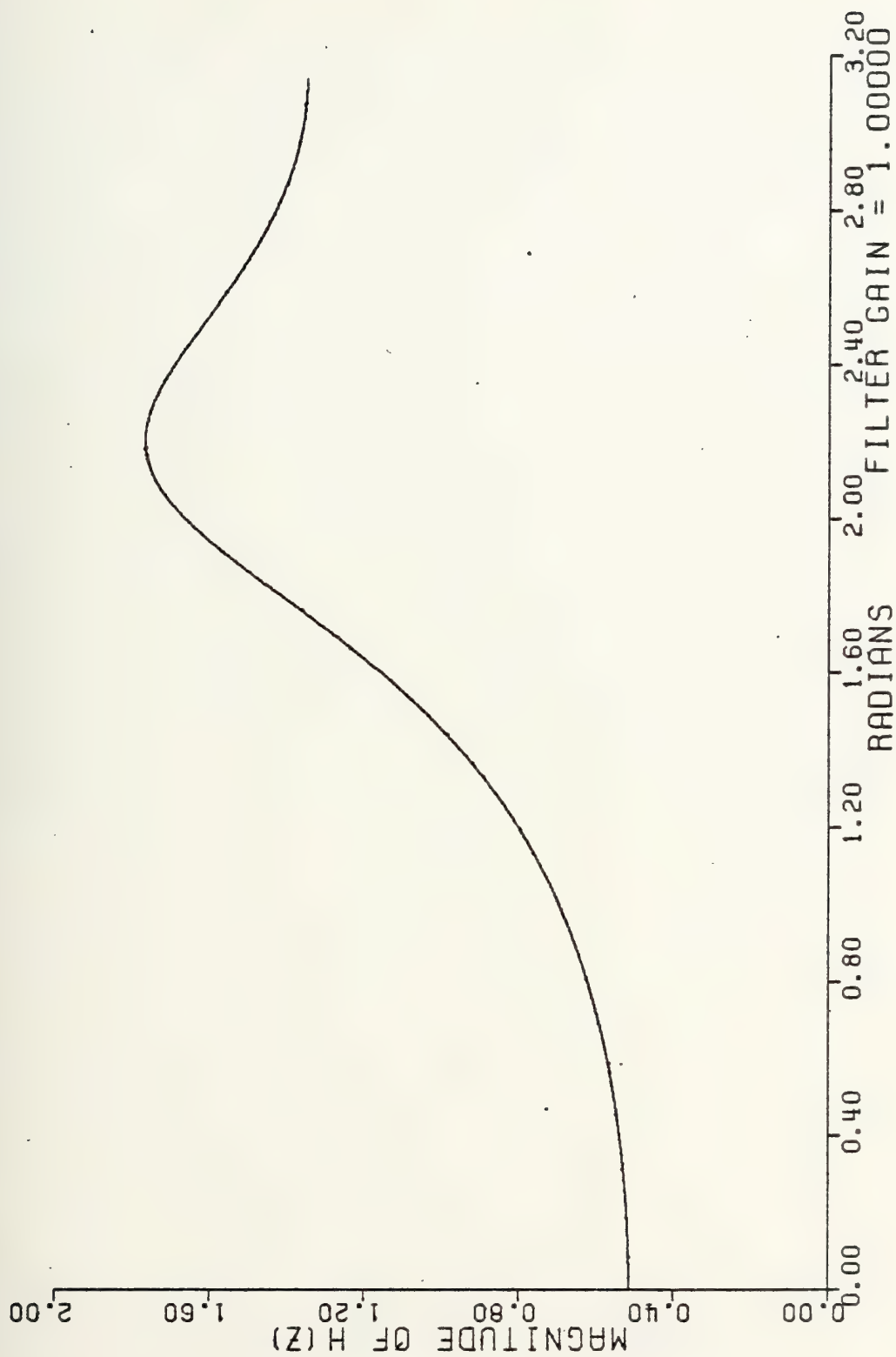


Figure 7-15e Magnitude Of  $H(z)$  For A Complex Pole Pair At  $(-0.3 + j0.5)$





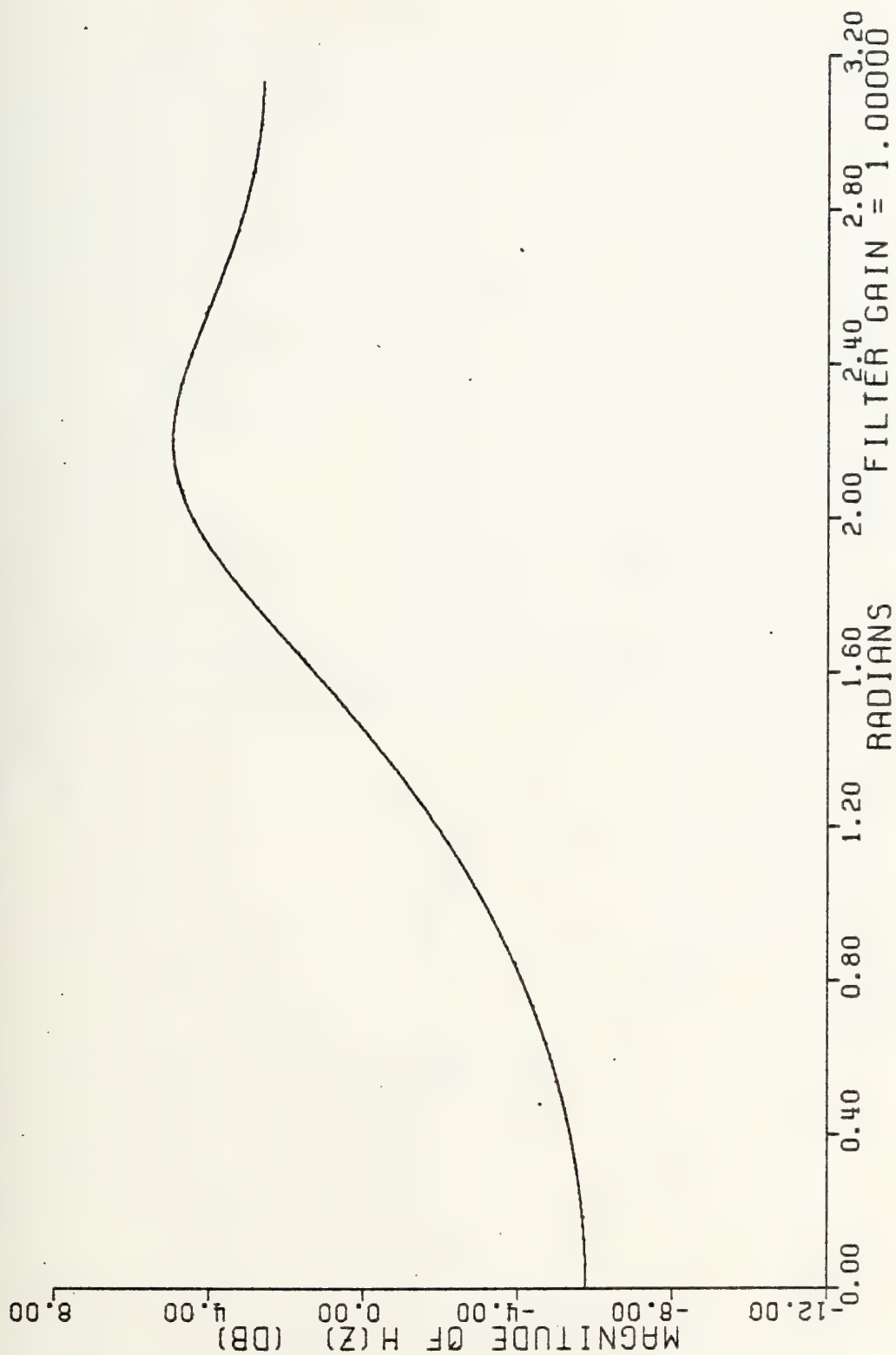


Figure 7-15f Magnitude Of (Hz) In Decibels For A Complex Pole Pair At  $(-0.3 + j0.5)$



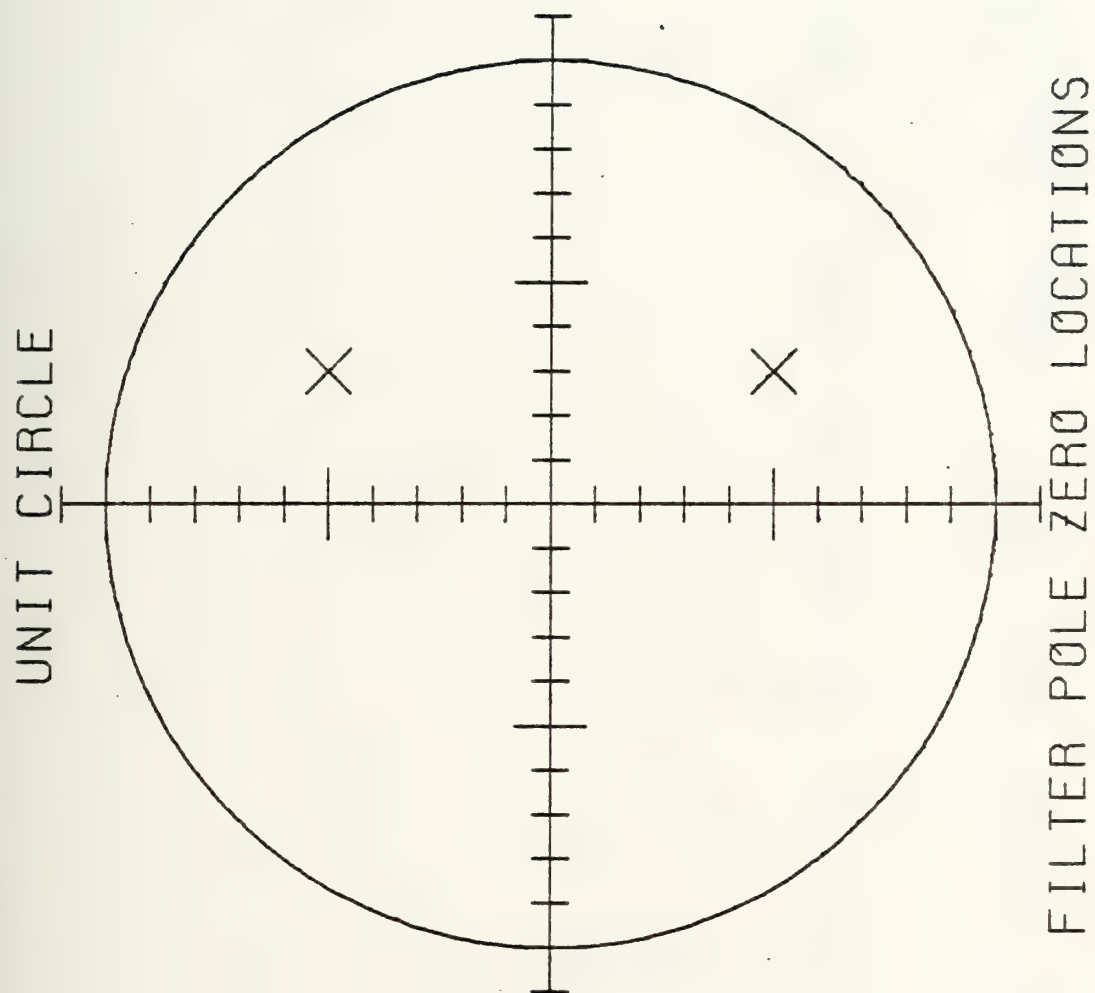


Figure 7-16a Example Of A Complex Pole Pair At  $(0.3 \pm j0.5)$



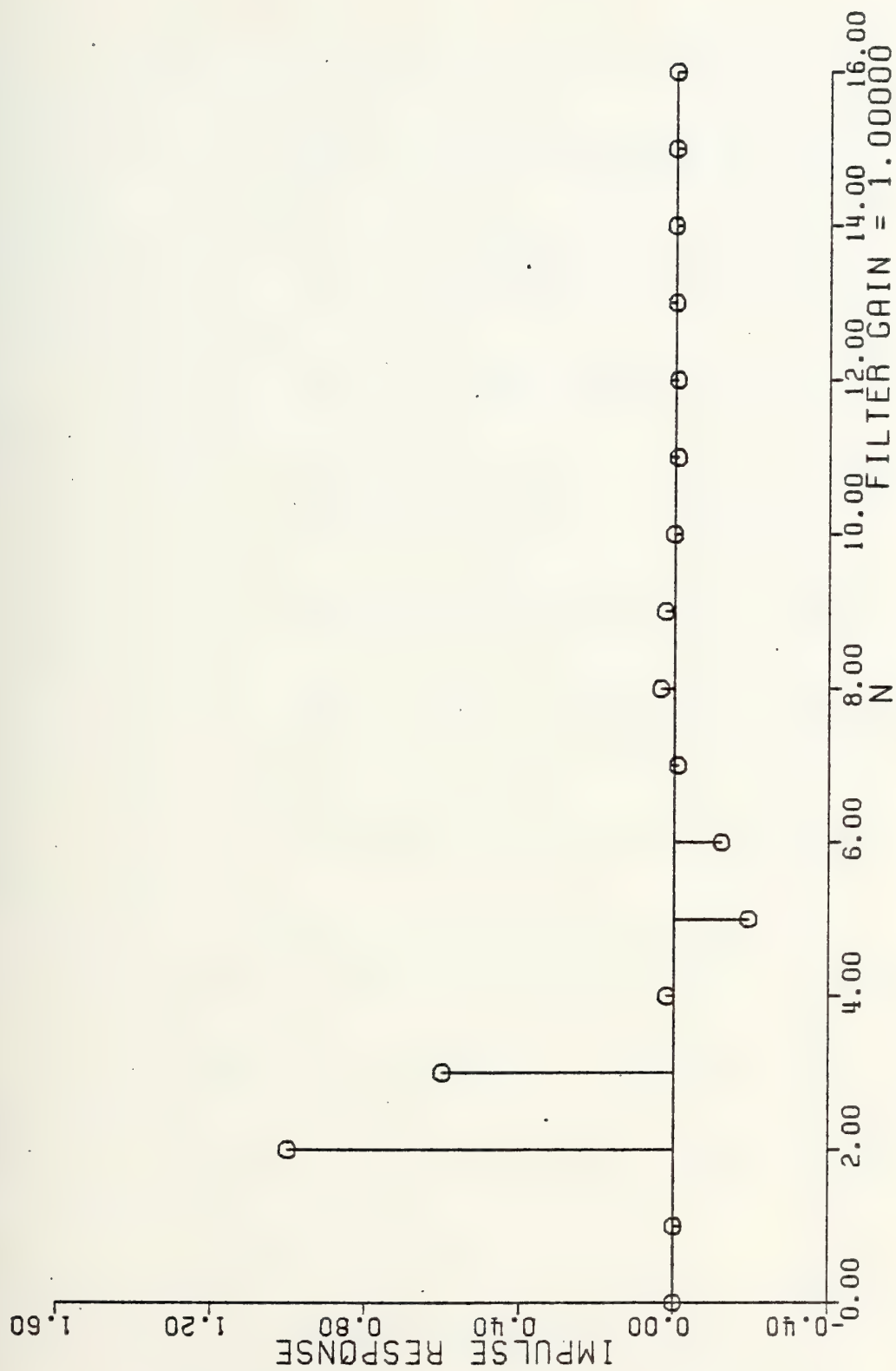


Figure 7-16b Unit Sample Response For A Complex Pole Pair At  $(0.3 + j0.5)$



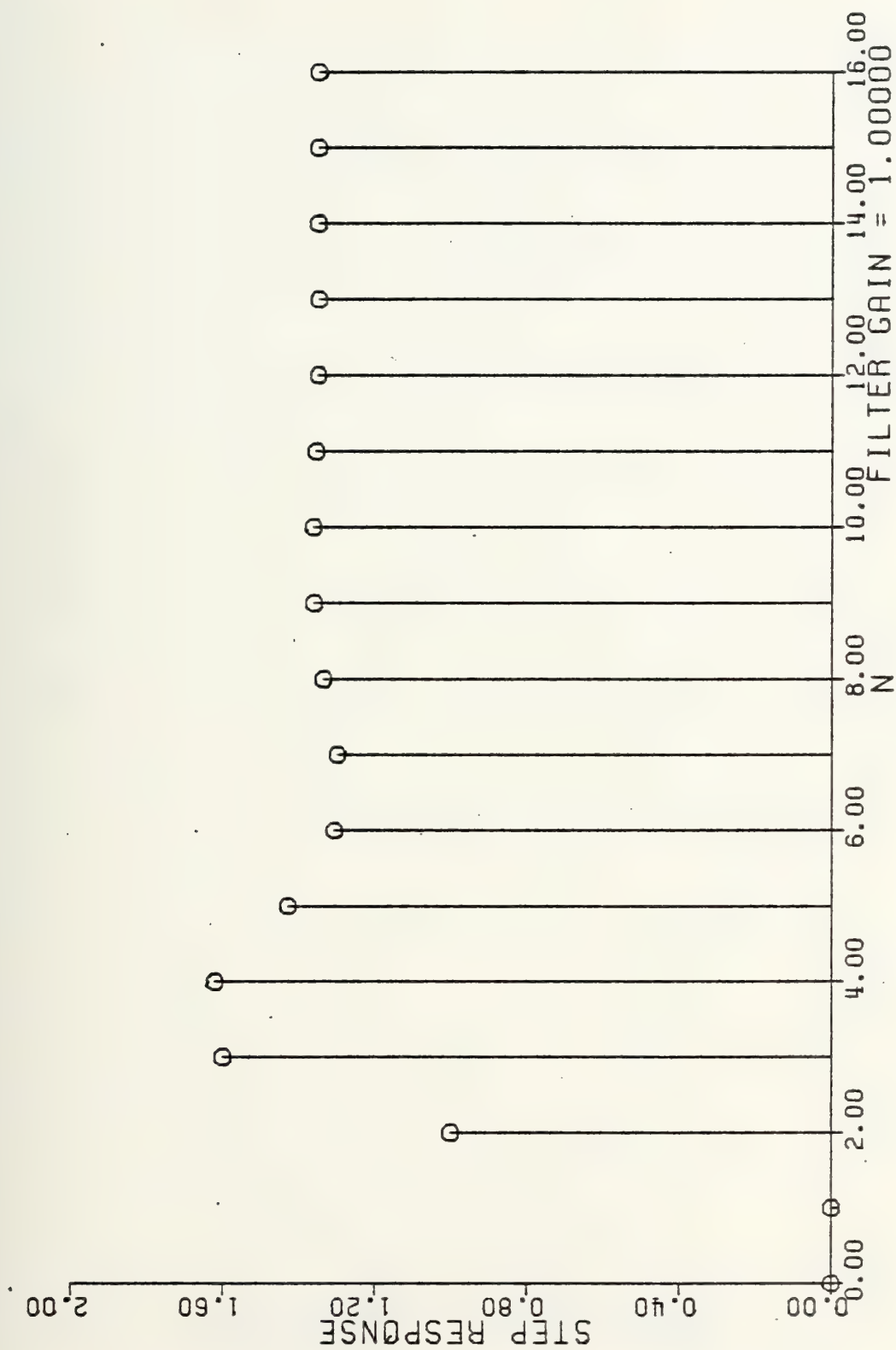


Figure 7-16c Unit Step Response For A Complex Pole Pair At  $(-0.3 + j0.5)$





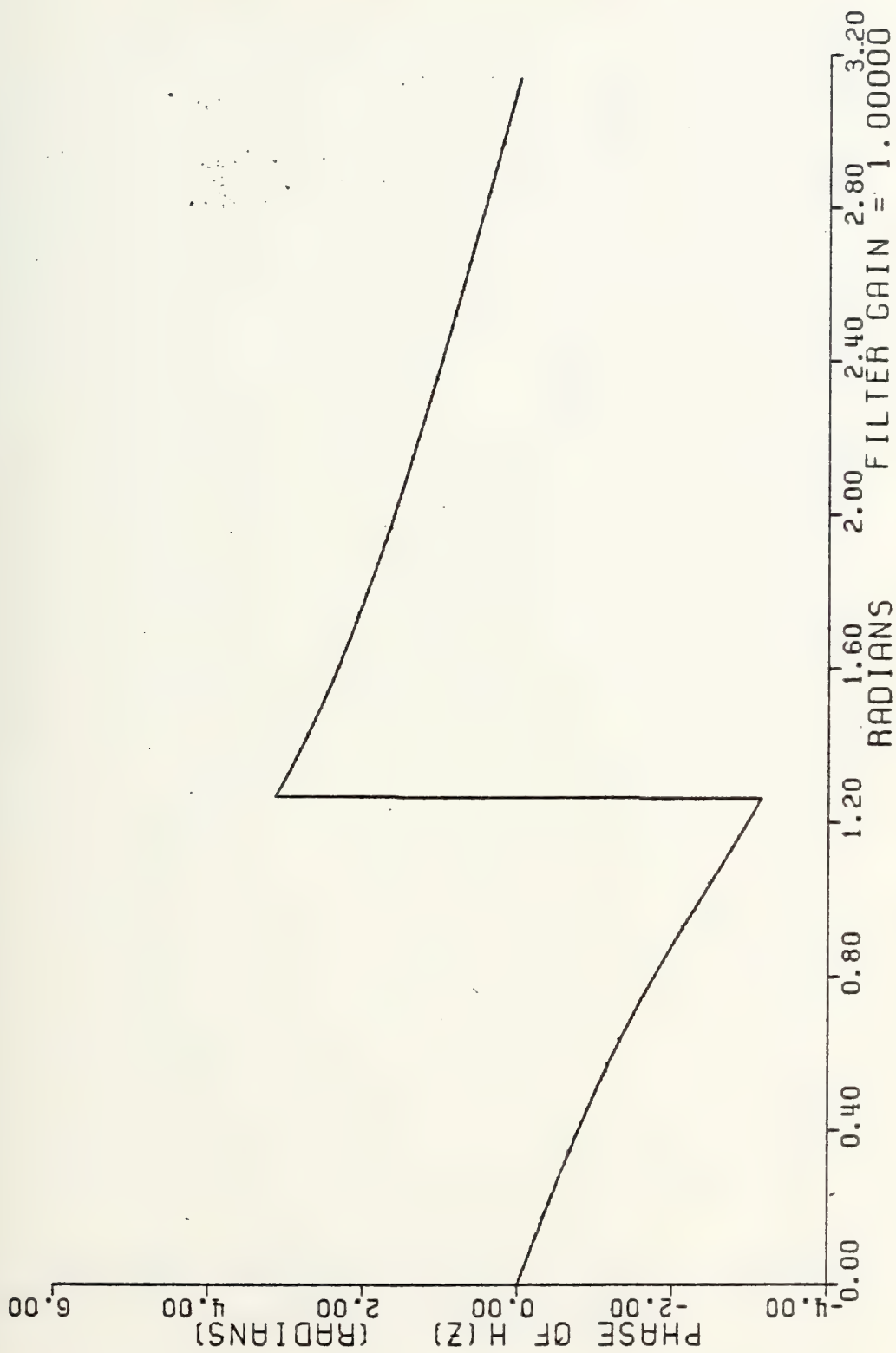


Figure 7-16d Phase Response For A Complex Pole Pair At  $(0.3 \pm j0.5)$



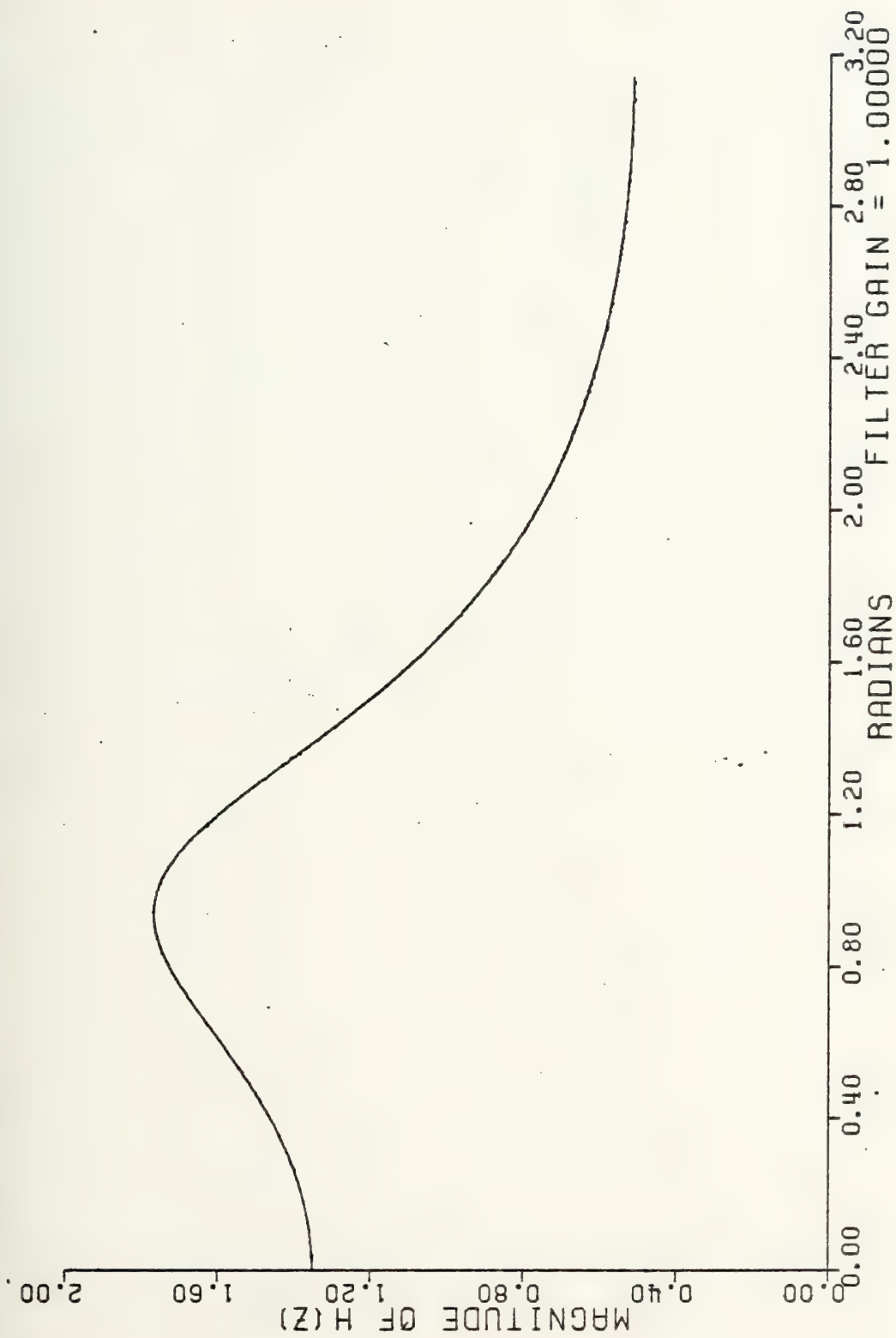


Figure 7-16e Magnitude Of  $H(z)$  For A Complex Pole Pair At  $(0.3 + j0.5)$



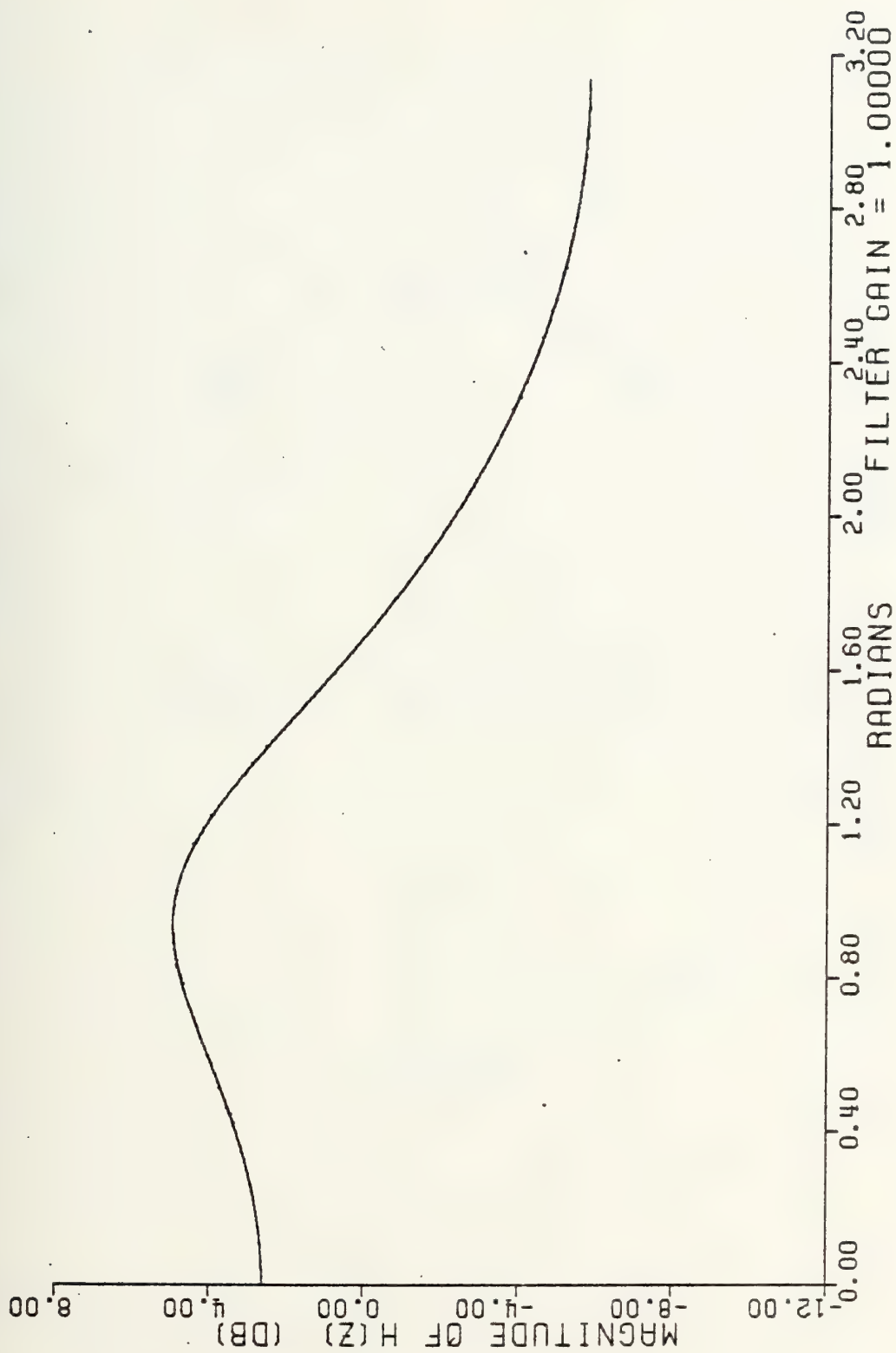


Figure 7-16f Magnitude Of  $H(z)$  In Decibels For A Complex Pole Pair At  $(0.3 + j0.5)$



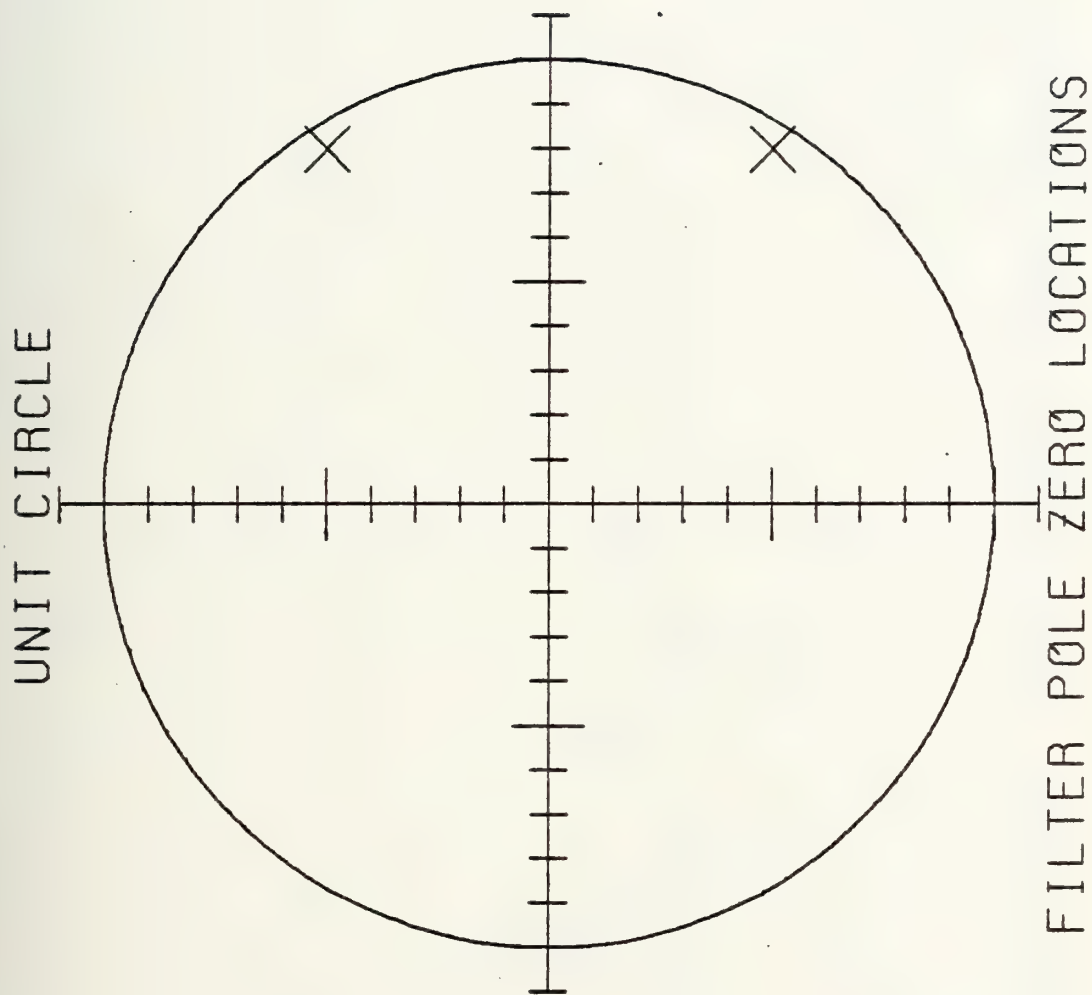


Figure 7-17a Example of A Complex Pole Pair At  $(0.8 \pm j0.5)$





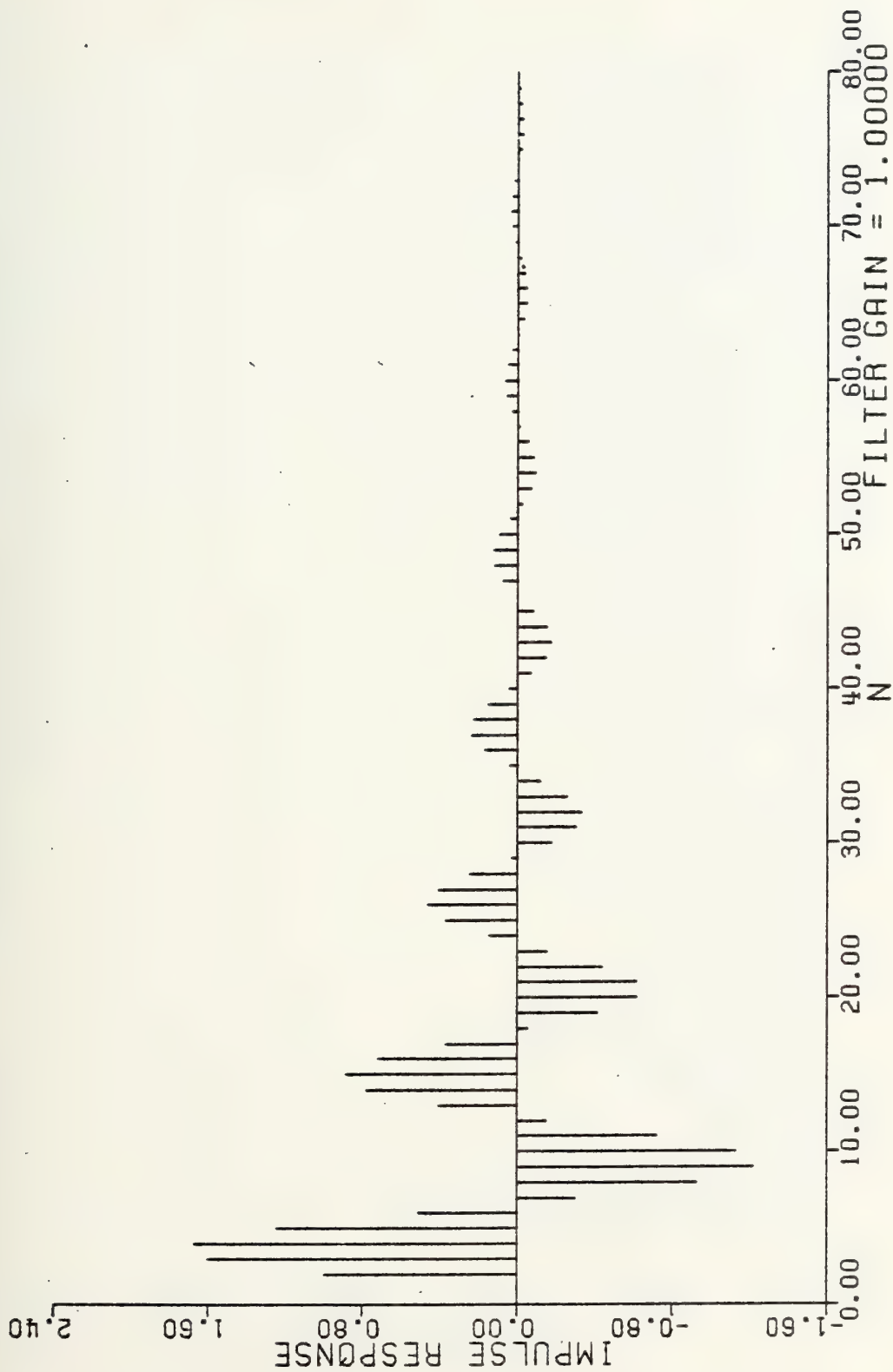


Figure 7-17b Unit Sample Response For A Complex Pole Pair At  $(0.8 + j0.5)$



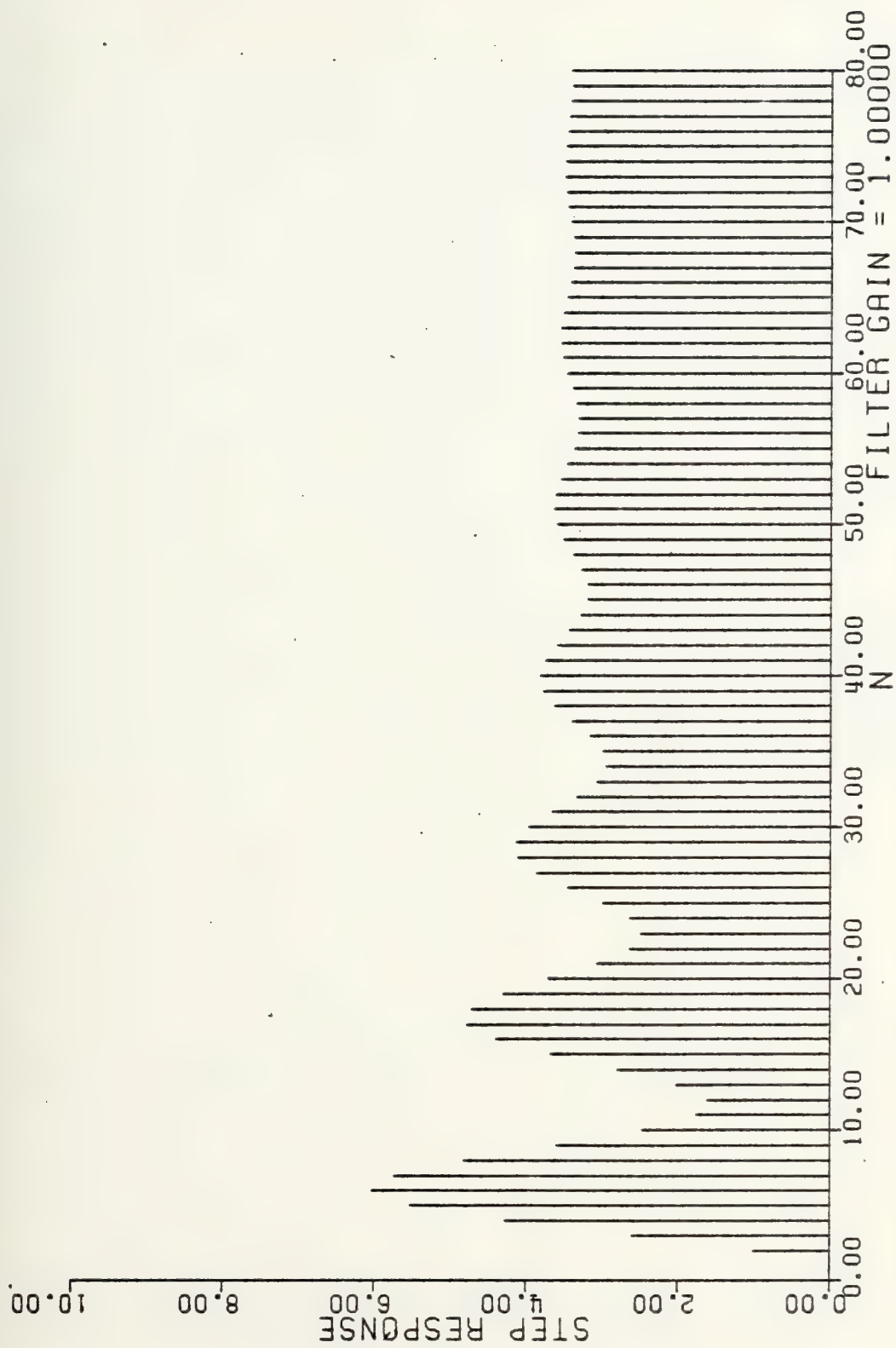


Figure 7-17c Unit Step Response For A Complex Pole Pair At  $(0.8 + j0.5)$



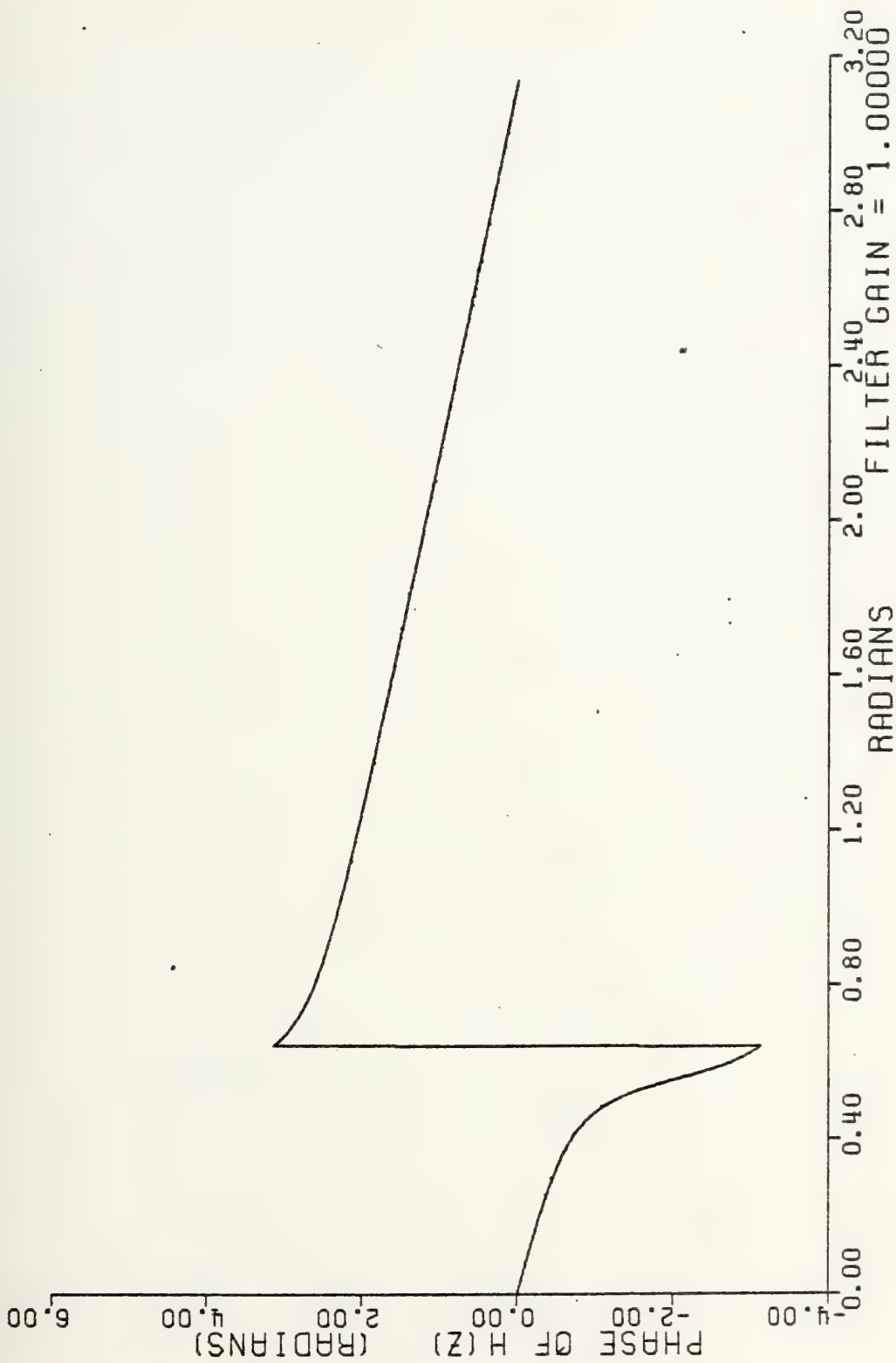


Figure 7-17d Phase Response For A Complex Pole Pair At  $(0.8 + j0.5)$



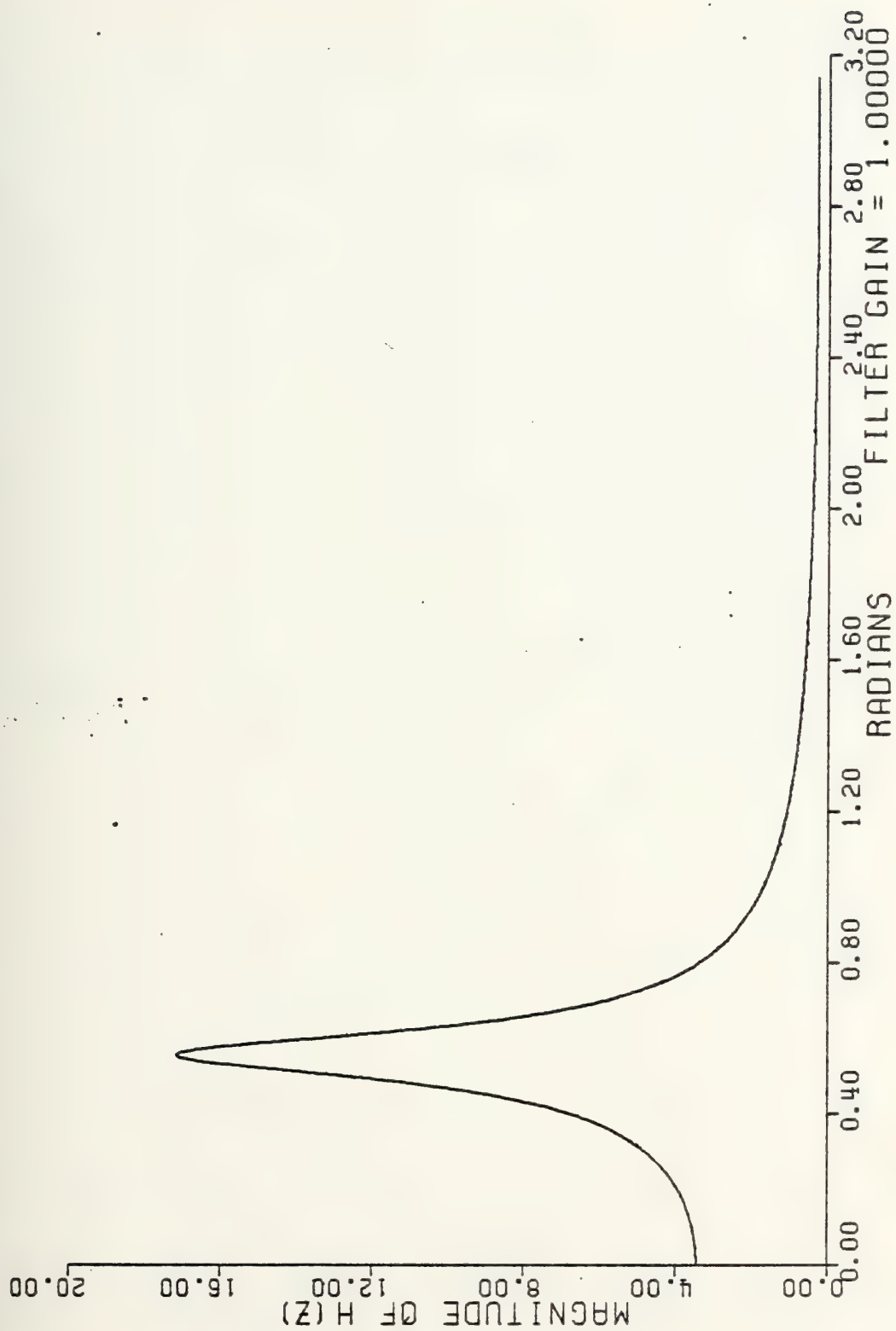


Figure 7-17e Magnitude Of  $H(z)$  For A Complex Pole Pair At  $(0.8 + j0.5)$





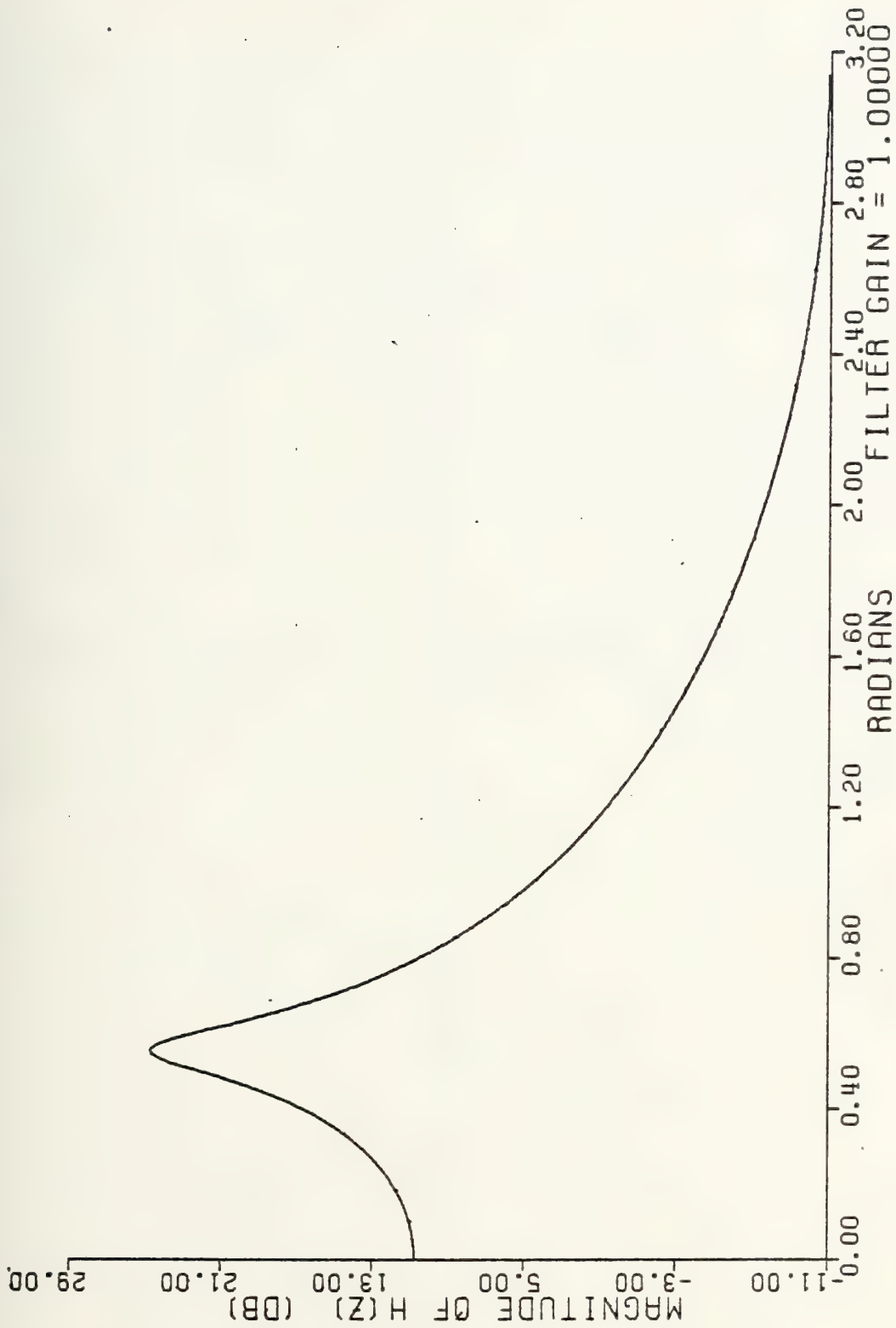


Figure 7-17f Magnitude Of  $H(z)$  In Decibels For A Complex Pole Pair At  $(0.8 + j0.5)$



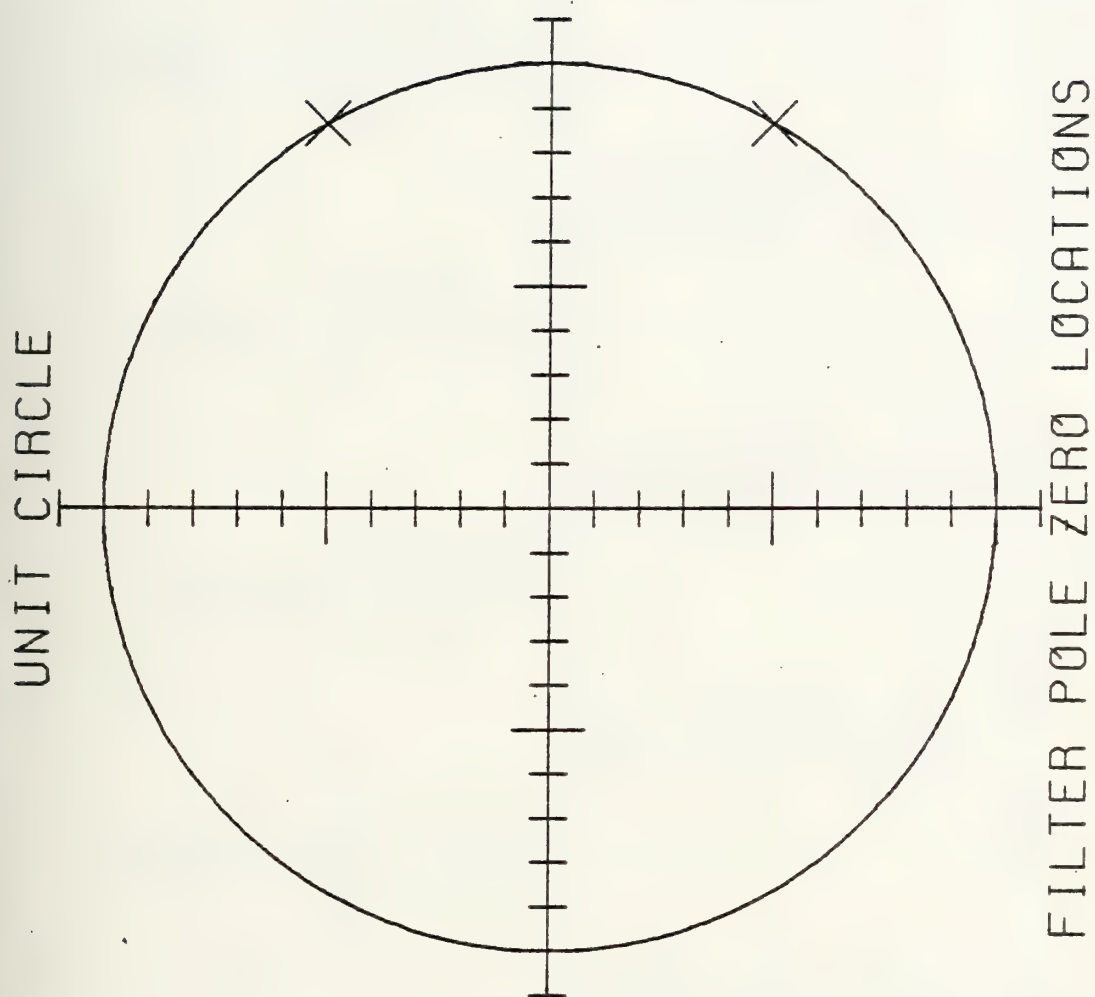


Figure 7.18a Example Of A Complex Pole Pair On The Unit Circle At  $(0.866025+j0.5)$



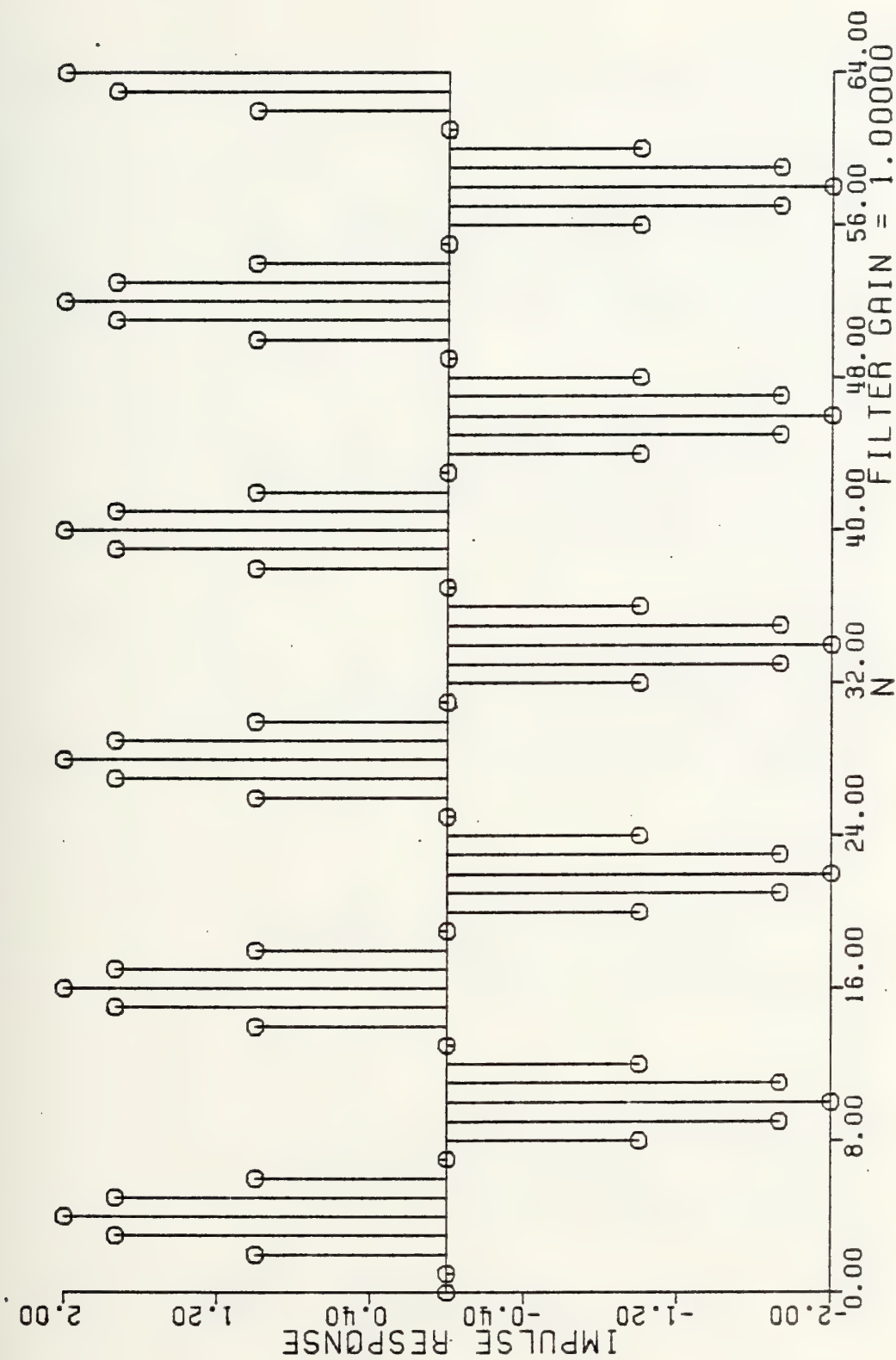


Figure 7-18b Unit Sample Response For A Complex Pole Pair At  $(0.866025 + j0.5)$



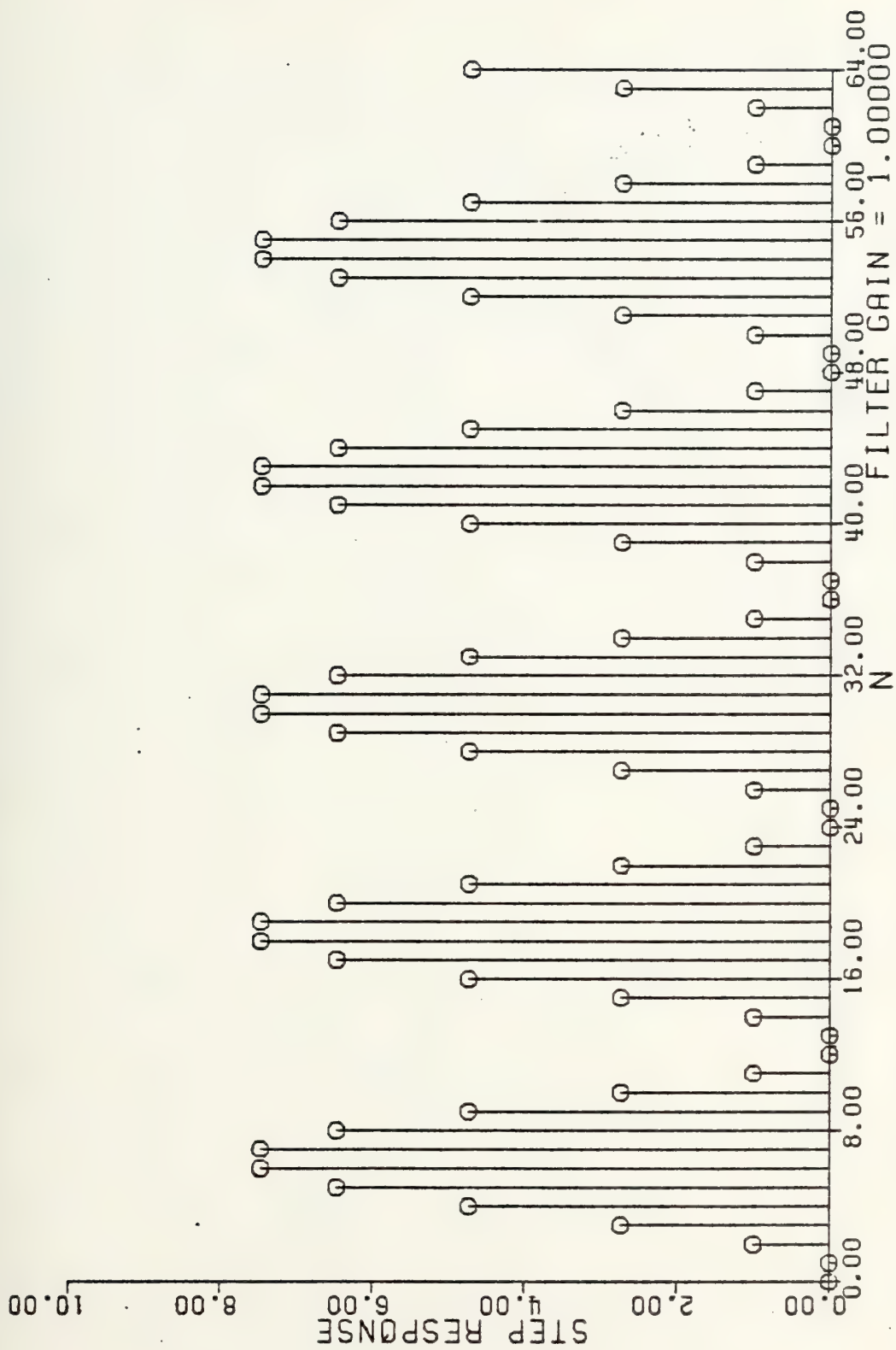


Figure 7-18c Unit Step Response For A Complex Pole Pair At  $(0.866025 + j0.5)$





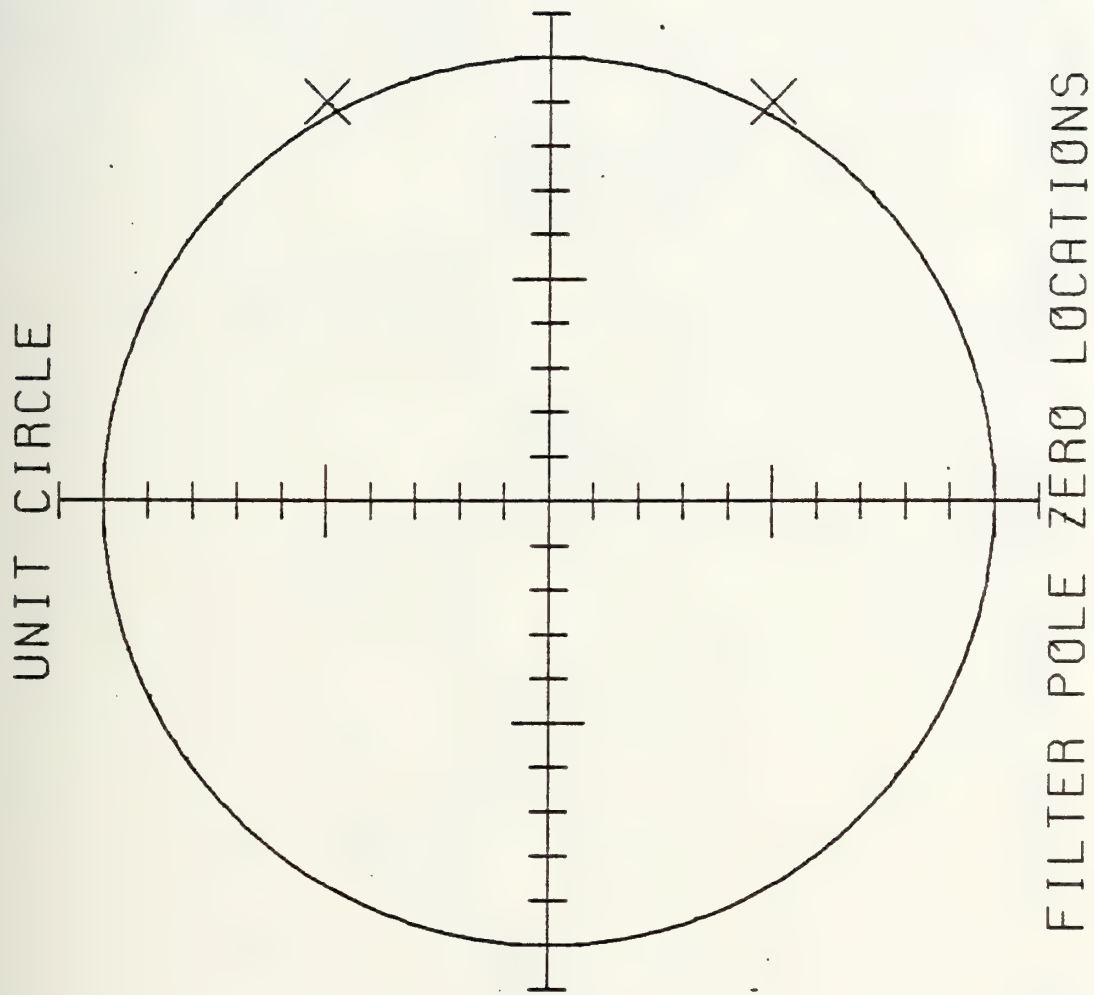


Figure 7-19a Example Of A Complex Pole Pair Outside The Unit Circle At  $(0.9 + j0.5)$



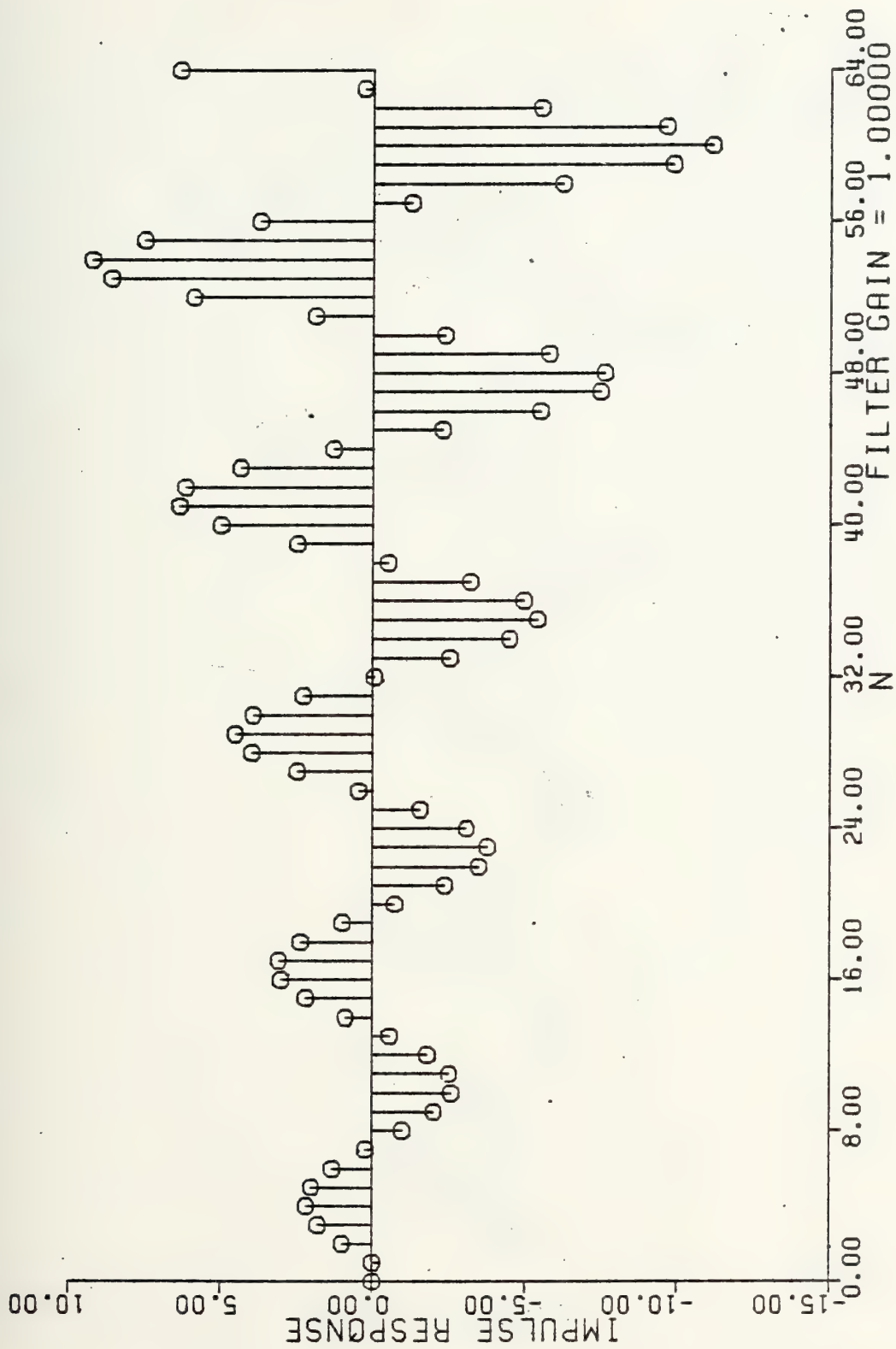


Figure 7-19b Unit Sample Response For A Complex Pole Pair At  $(0.9 + j0.5)$



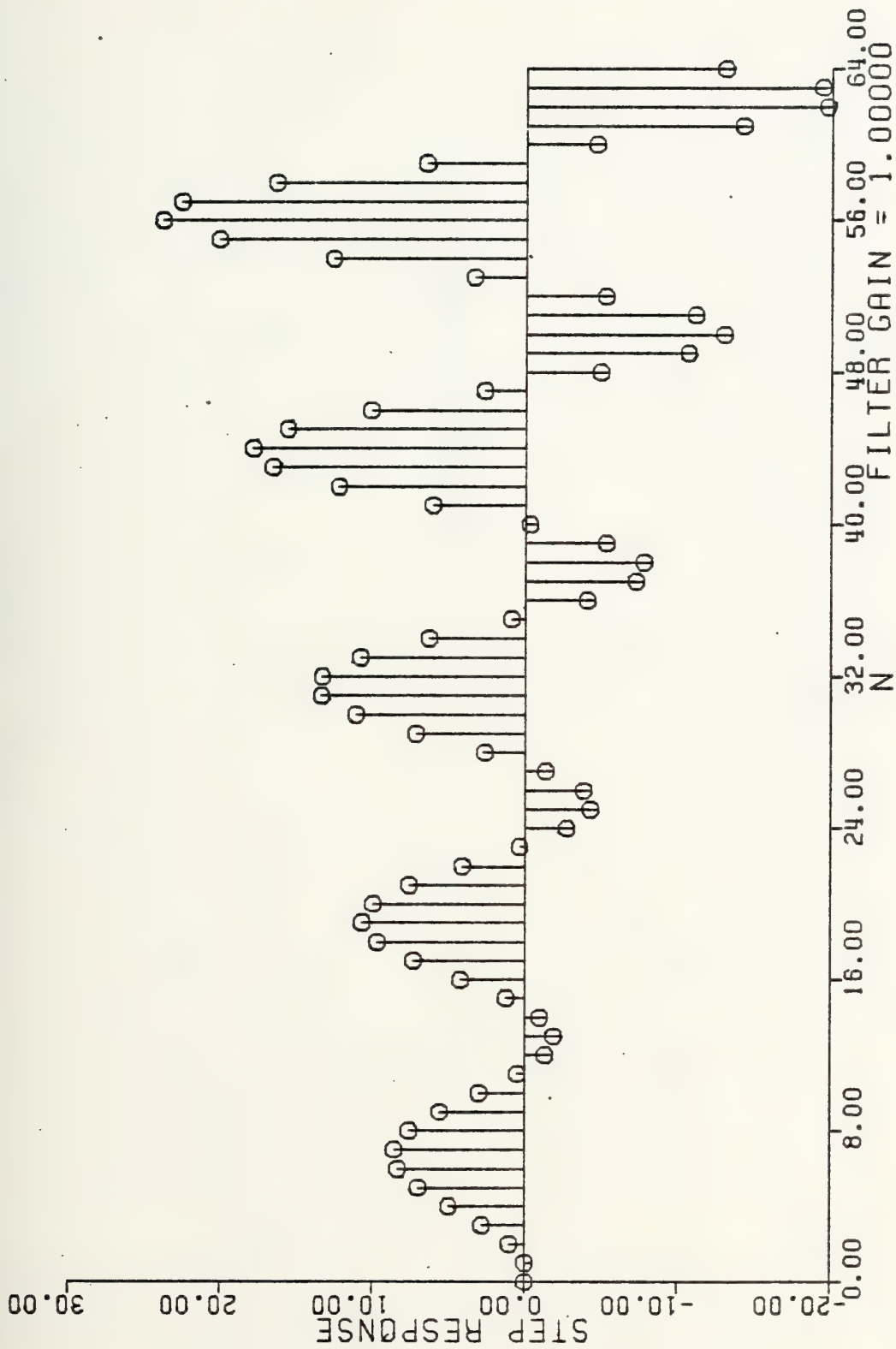


Figure 7-19c Unit Step Response For A Complex Pole Pair At  $(0.9 + j0.5)$



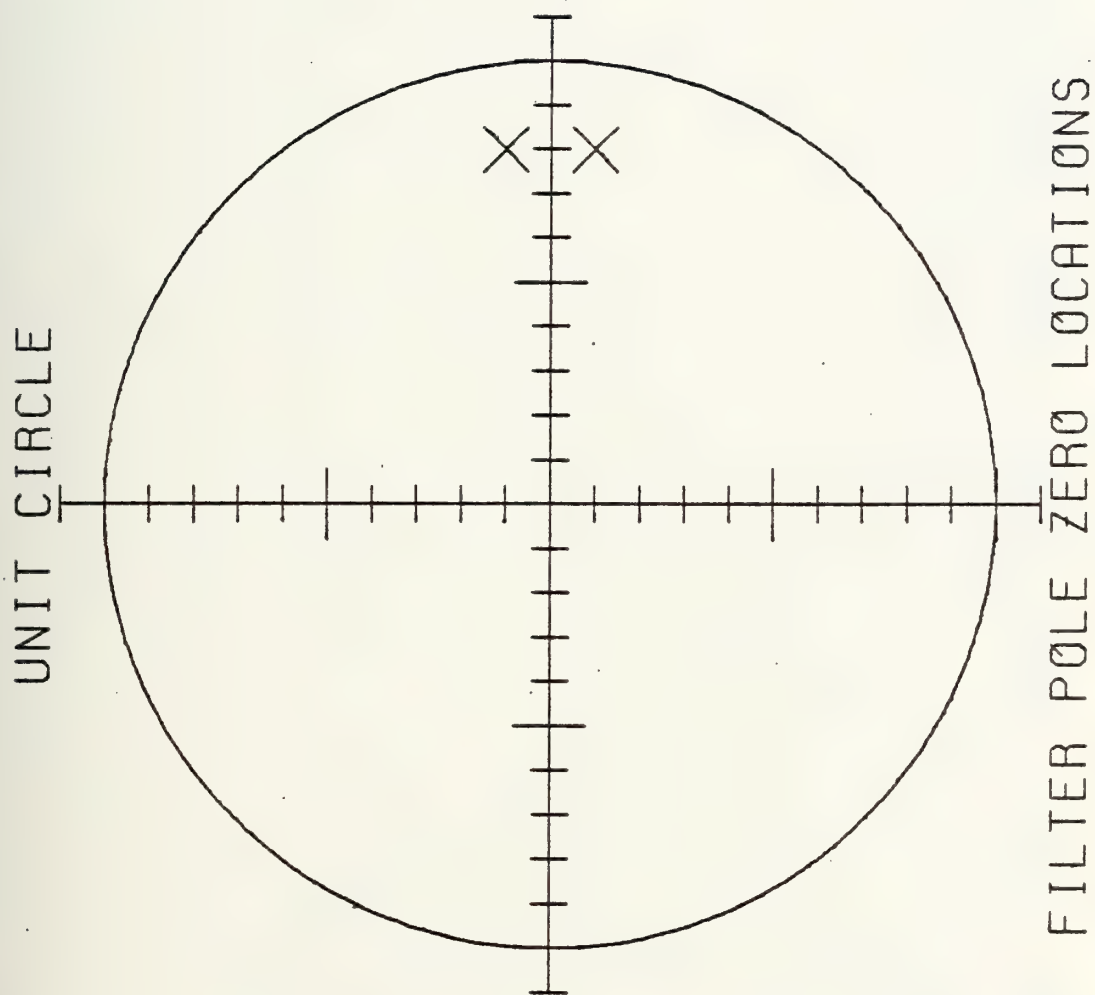


Figure 7-20a Example Of A Complex Pole Pair At  $(0.8 \pm j0.1)$





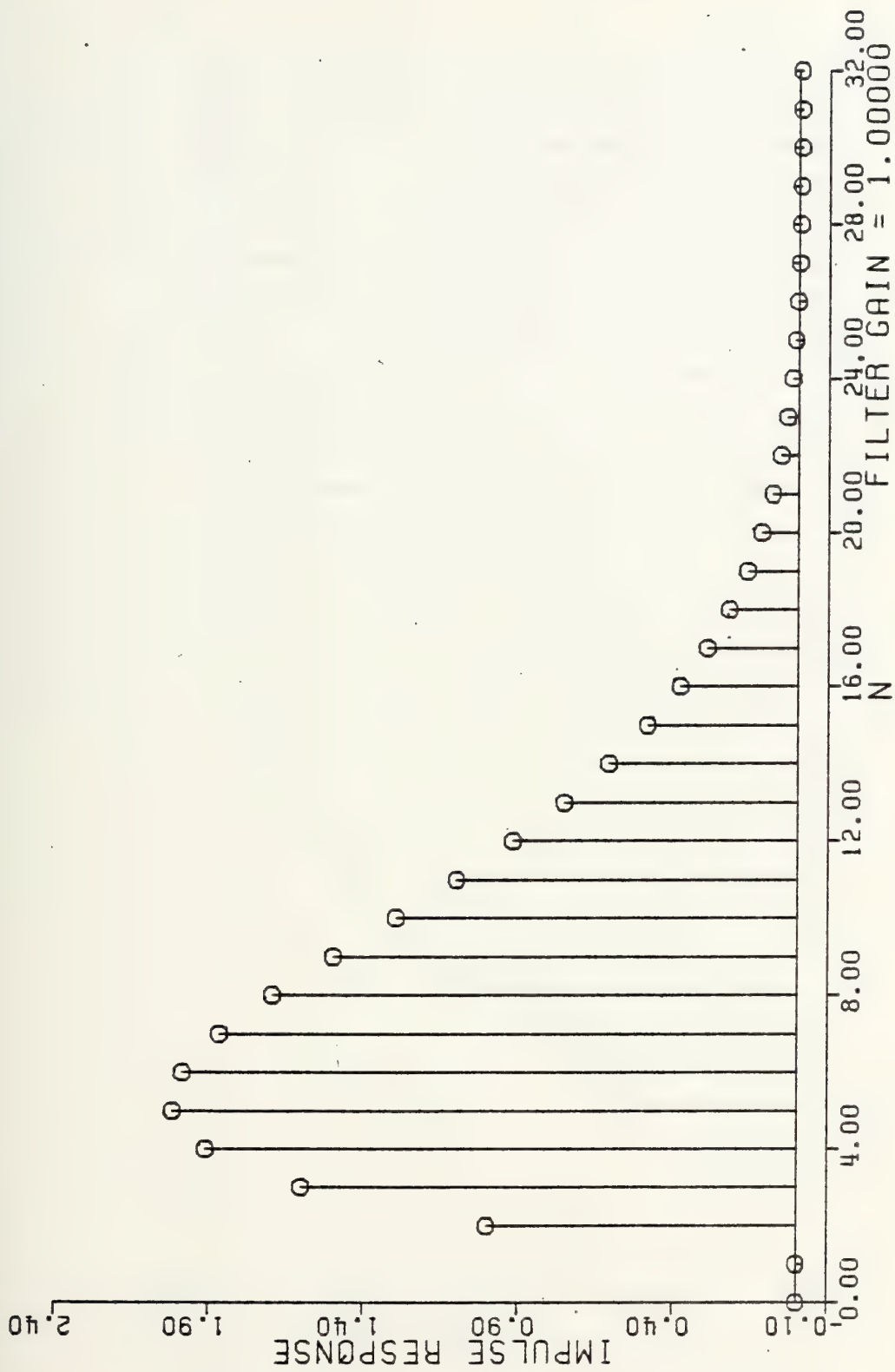


Figure 7-20b Unit Sample Response Of A Complex Pole Pair At  $(0.8 + j0.1)$



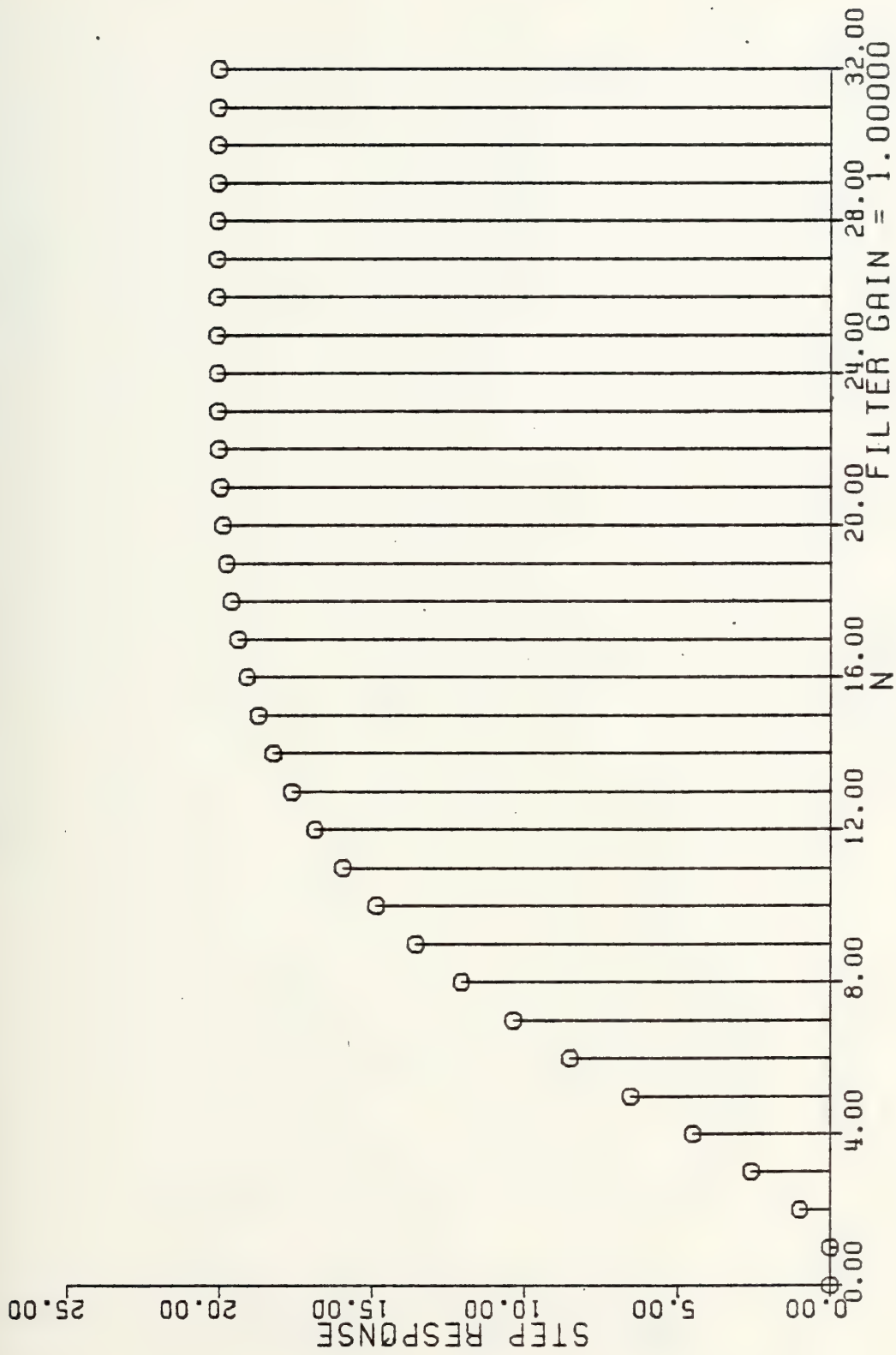


Figure 7-20c Unit Step Response For A Complex Pole Pair At  $(0.8 + j0.1)$



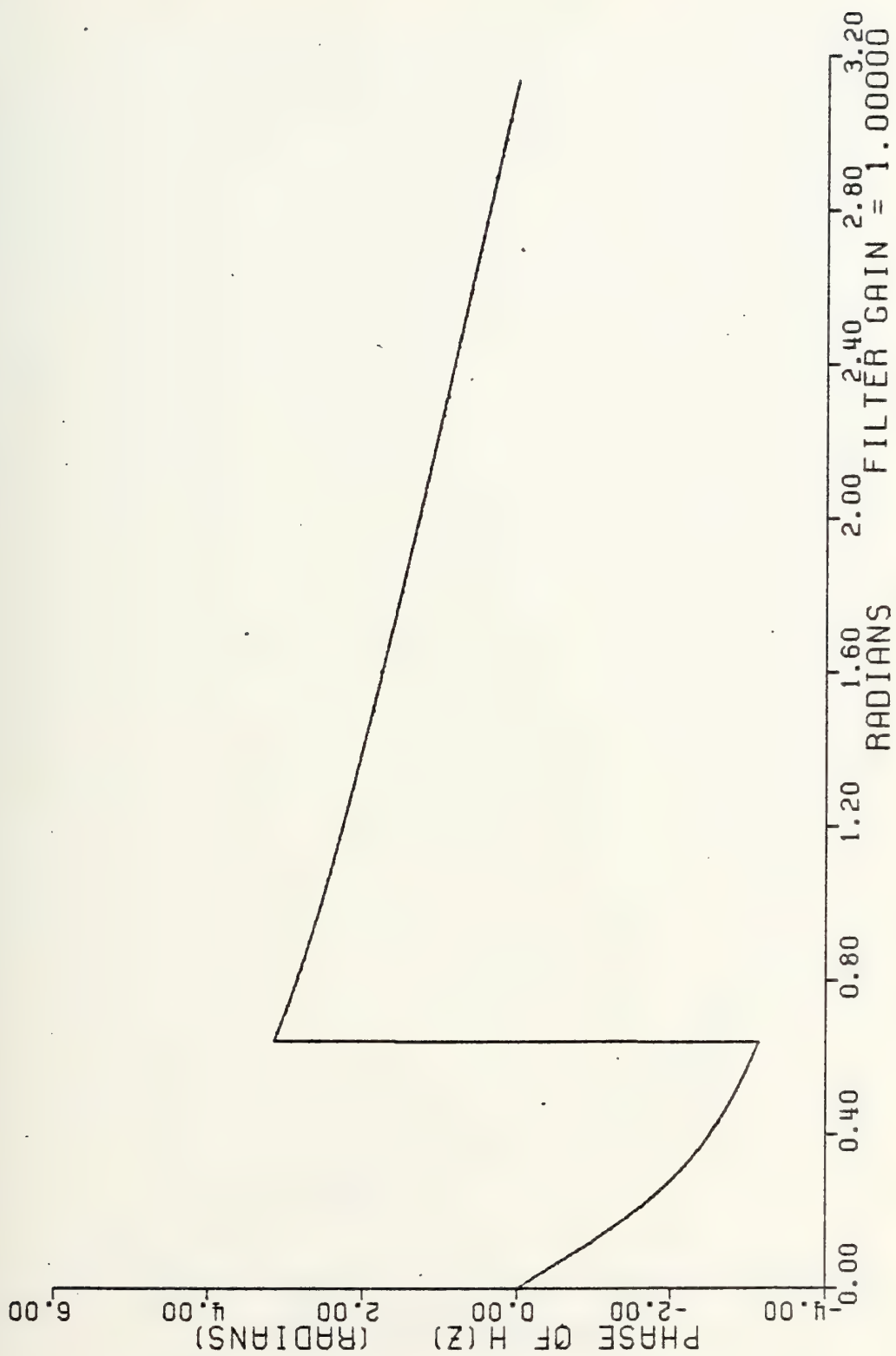


Figure 7-20d Phase Response For A Complex Pole Pair At  $(0.8 + j0.1)$



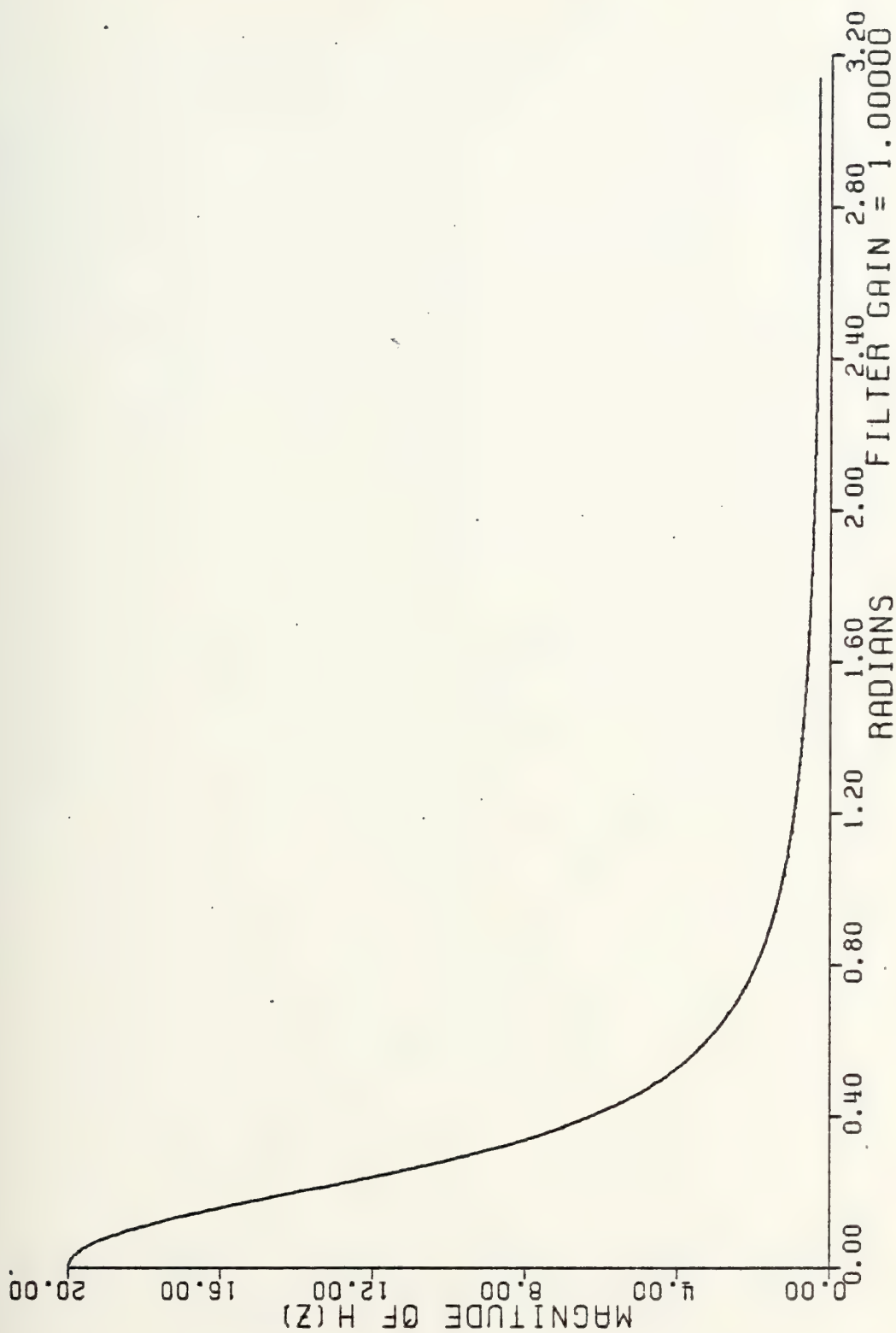


Figure 7-20e Magnitude Of  $H(z)$  For A Complex Pole Pair At  $(0.8 + j0.1)$





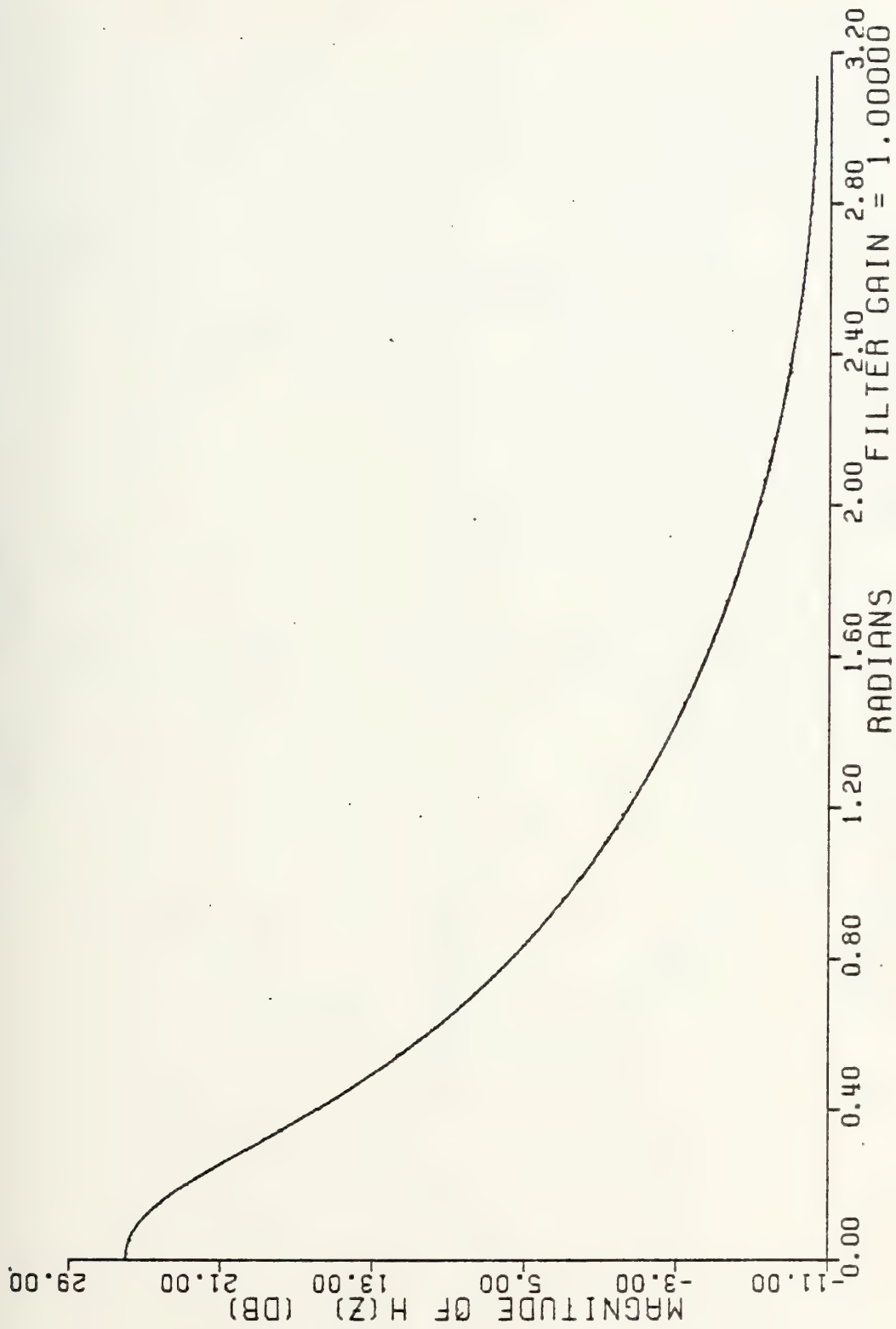


Figure 7-20f Magnitude Of  $H(z)$  In Decibels For A Complex Pole Pair At  $(0.8 + j0.1)$



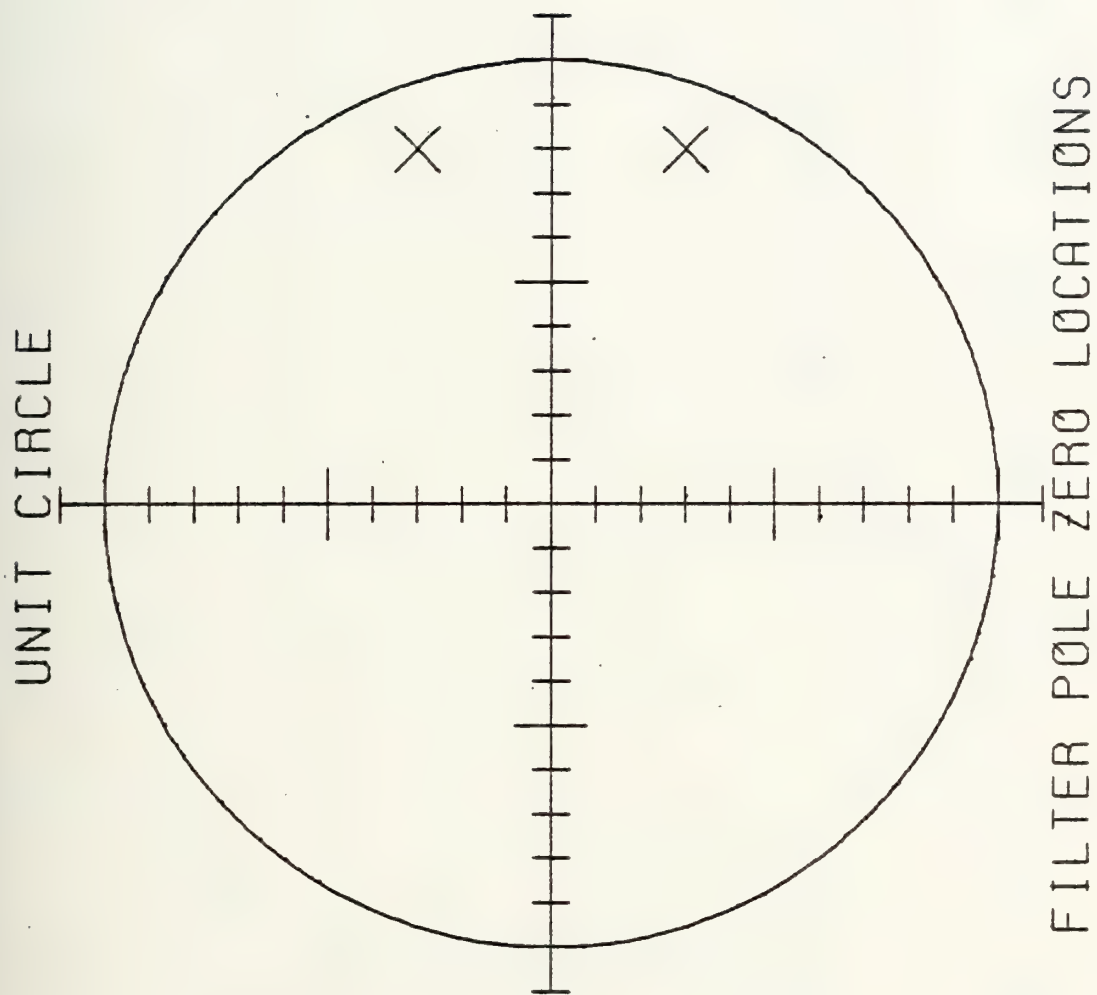


Figure 7-21a Example Of A Complex Pole Pair At  $(0.8 \pm j0.3)$



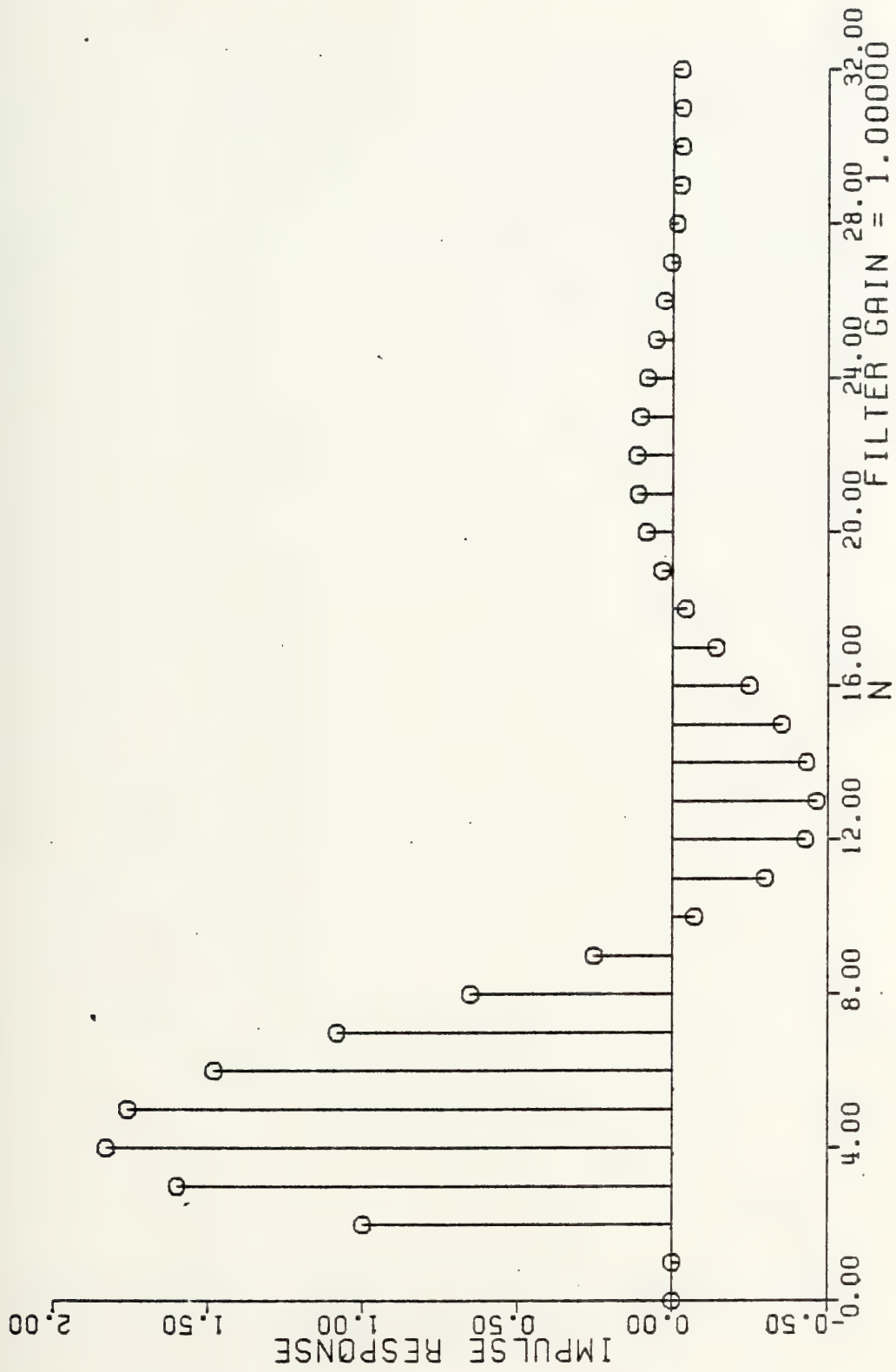


Figure 7-21b Unit Sample Response For A Complex Pole Pair At  $(0.8 + j0.3)$



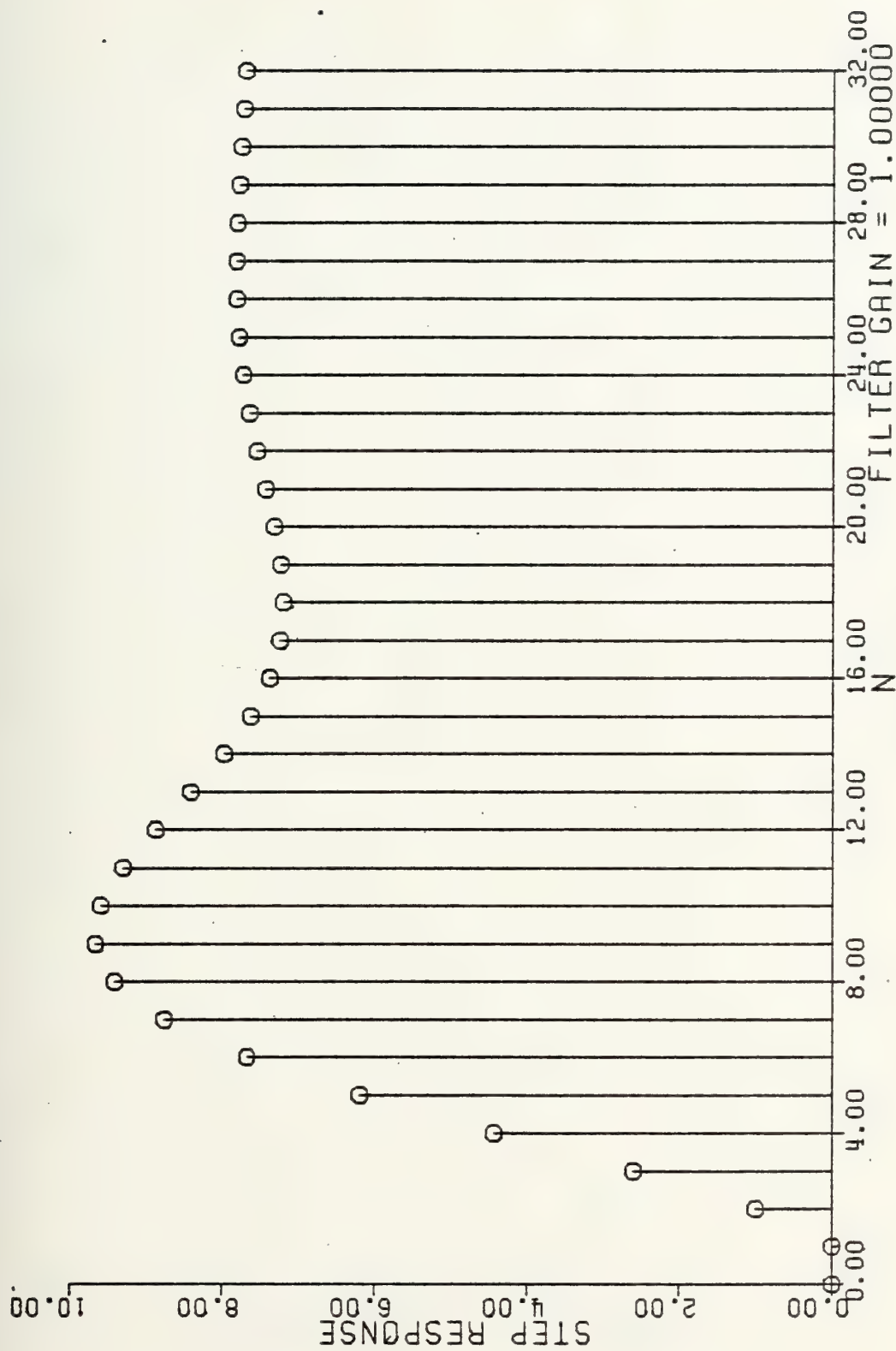


Figure 7-21c Unit Step Response For A Complex Pole Pair At  $(0.8 \pm j0.3)$





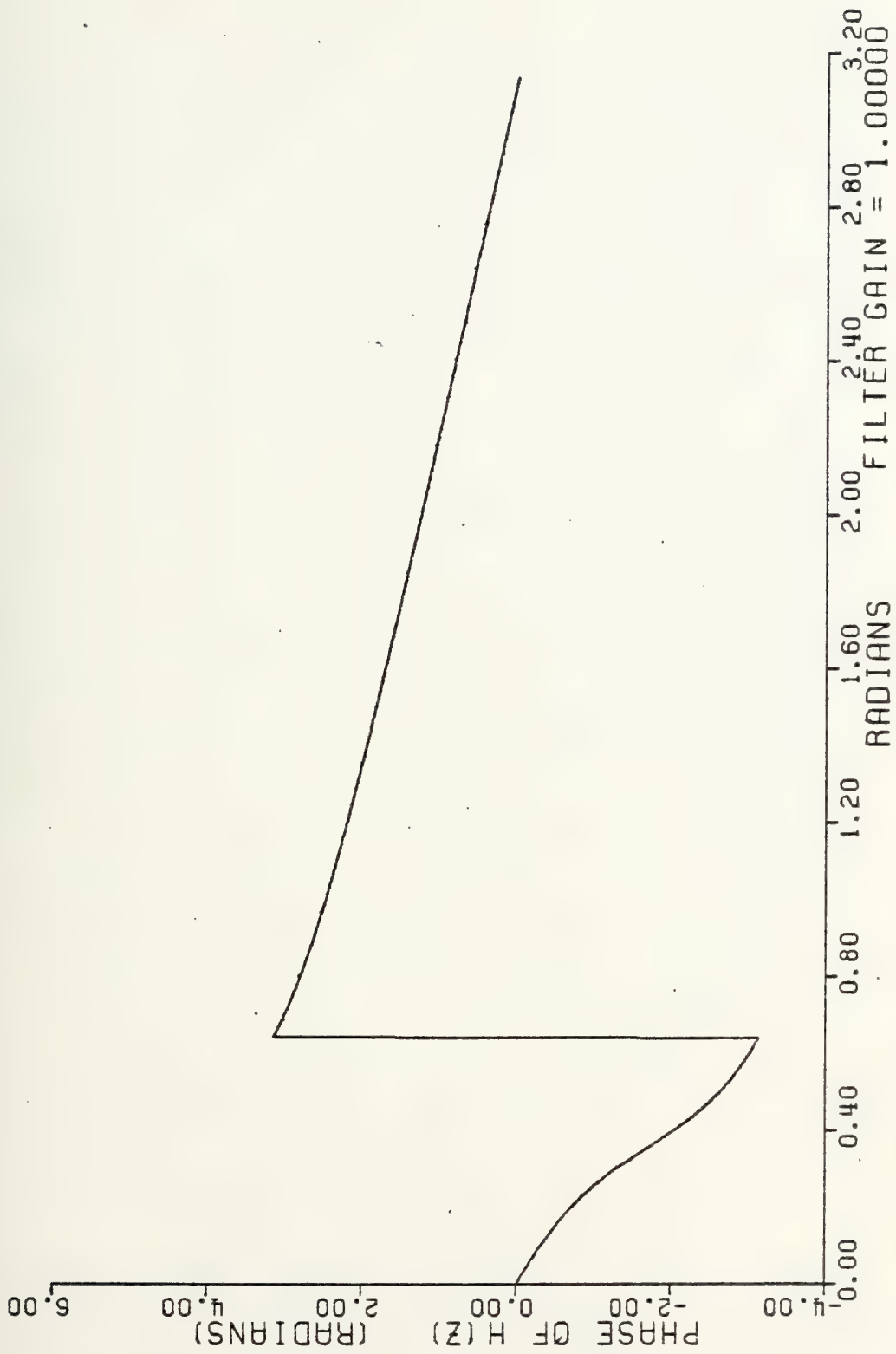


Figure 7-2ld Phase Response For A Complex Pole Pair At  $(0.8 + j0.3)$



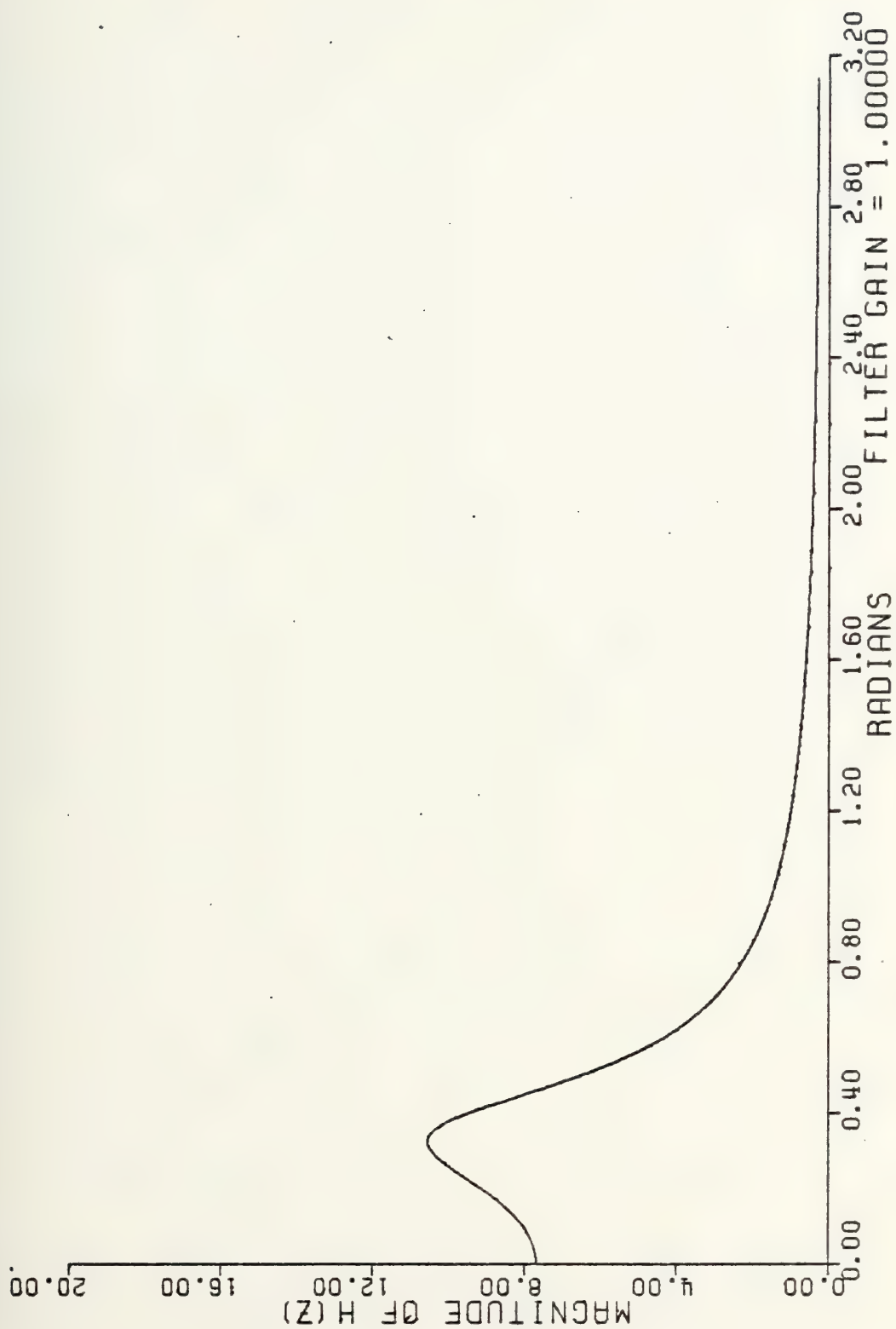


Figure 7-21e Magnitude Of  $H(z)$  For A Complex Pole Pair At  $(0.8 + j0.3)$



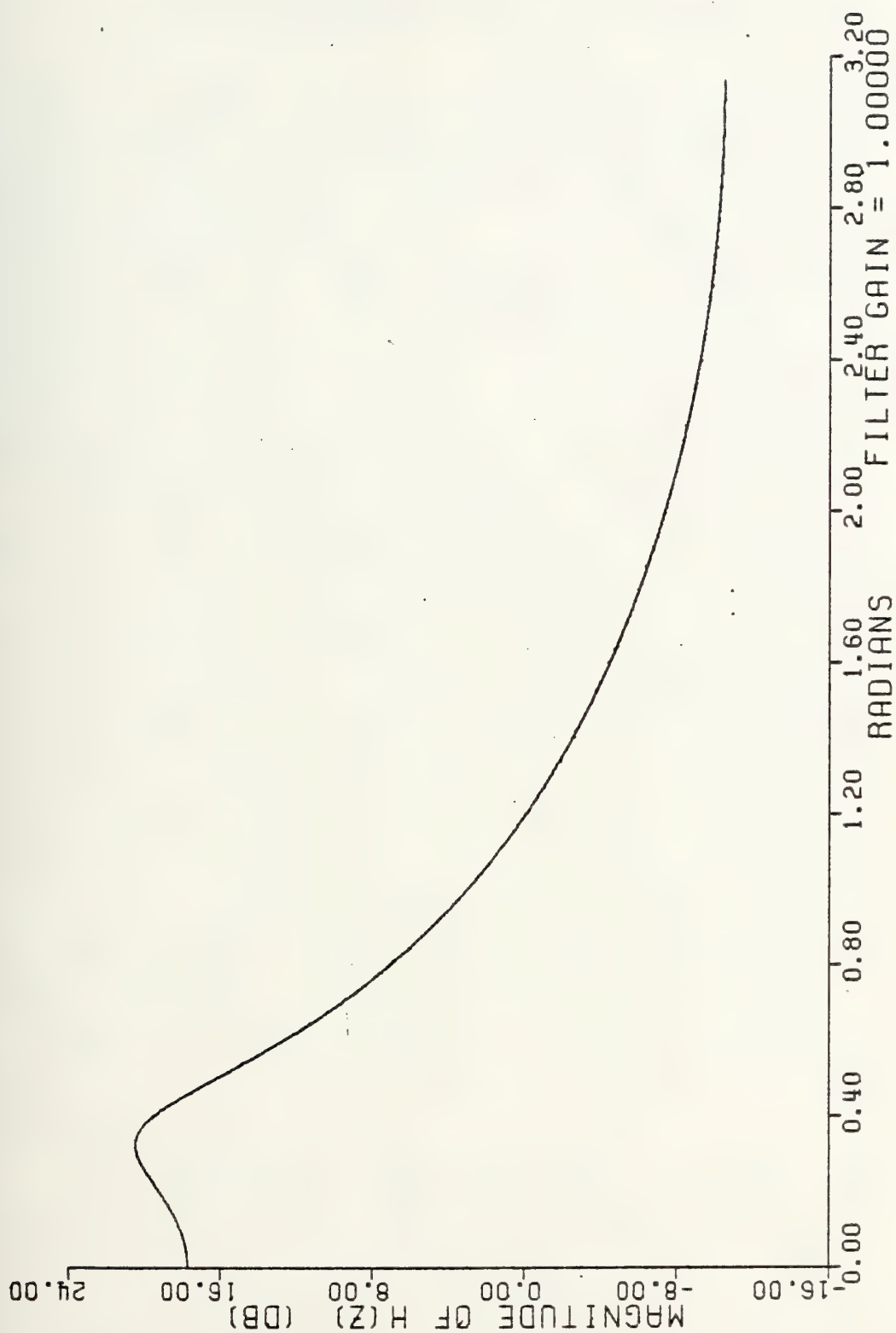


Figure 7-21f Magnitude Of  $H(z)$  In Decibels For a Complex Pole Pair At  $(0.8 + j0.3)$



## VIII. INTERPRETATION OF ASDF OUTPUT

When the user initially confronts the ASDF program it is expected that he will attempt to duplicate some of the examples presented earlier in this thesis or similar examples found in the literature. These types of applications of ASDF are straightforward and the results generated can be compared with published results. Interpretation of the output plots is seldom confusing.

As the user becomes more familiar with the use of the ASDF program and the positioning of the poles and zeros in the  $z$ -plane he will undoubtedly attempt to generate filters with sharper transition regions, and greater attenuation in the stop band(s). In general these filter improvements are accomplished by increasing the order of the system and positioning the poles closer to the unit circle. As the order of the system increases, and as the poles of the filter become closer to the unit circle, the filter becomes increasingly subject to the difficulties associated with the finite precision arithmetic used in the filter implementation.

A tacit assumption made very early in most digital signal processing courses (and often quickly forgotten) is that infinite precision arithmetic is available in performing the calculations. This assumption breaks down





when digital filters are realized in hardware or simulated! In general there are three sources of error which may affect the performance of digital filters. Due to the finite precision representation of the coefficients, the values of the implemented filter coefficients are only approximately equal to the desired values. Second, errors may be generated in the arithmetic operations. In general, the product of two  $n$ -bit numbers results in a  $2n$ -bit result. An error occurs when this  $2n$ -bit number is converted to  $n$ -bits for succeeding computations. The third type of error encountered is the analog to digital conversion required for processing analog signals with digital filters. Only the first two of these errors is of concern in the ASDF program.

While an in depth analysis of finite precision effects is beyond the scope of this work, let us consider the filter which has eight zeros located at the origin, and eight poles located at  $-0.9$ . Entering these locations into the  $z$ -plane using the ASDF program generates the filter responses in figures 6-1a, b, c. A superficial glance at these responses indicates that this filter is unstable. The filter must be stable, however, since all of the poles lie within the unit circle.

There are several phenomena involved in this example. The first is the dynamic range limitation of the system on which the filter is implemented. The IBM-360 is limited to computing with numbers between  $16^{64}$  and  $-16^{-63}$ . Once the



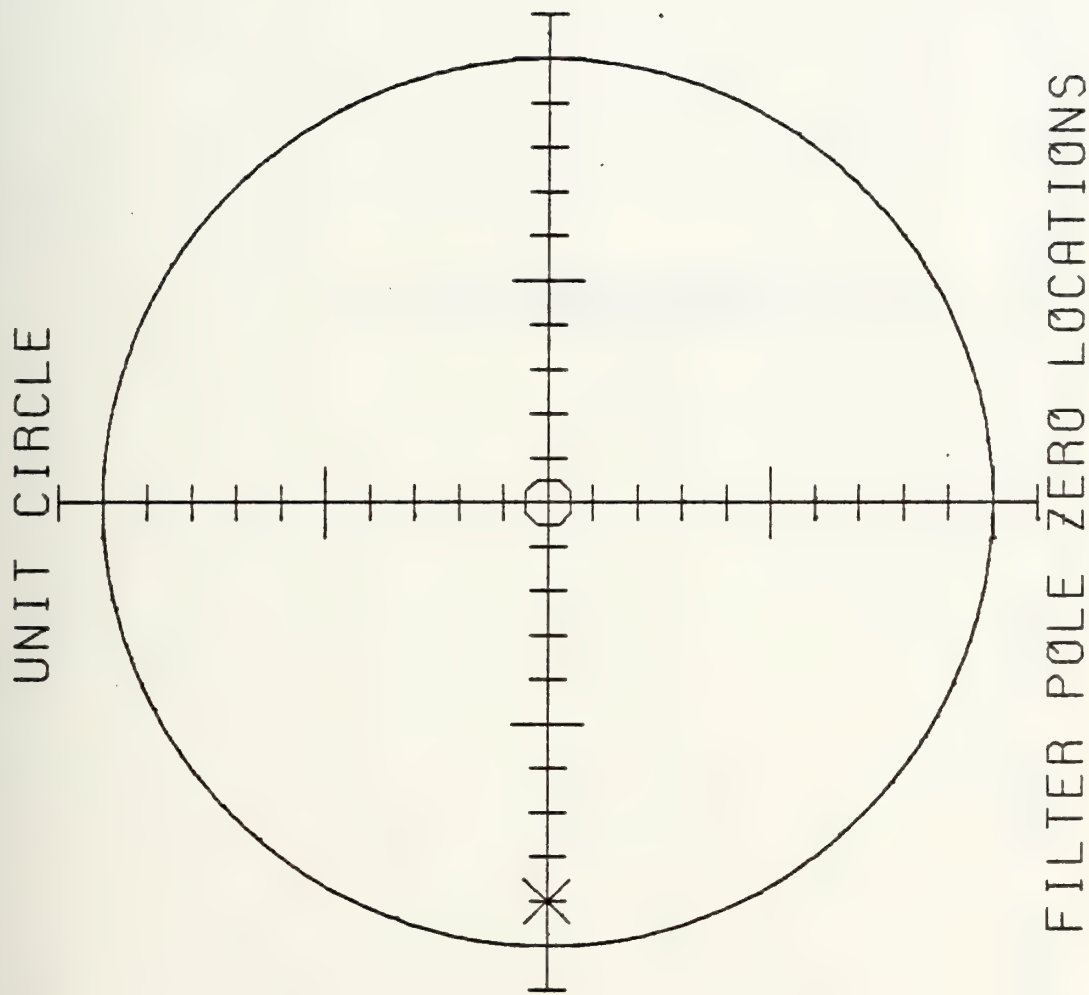


Figure 8-1a Example Filter With Eight Poles At  $-0.9$  And Eight Zeros At The Origin



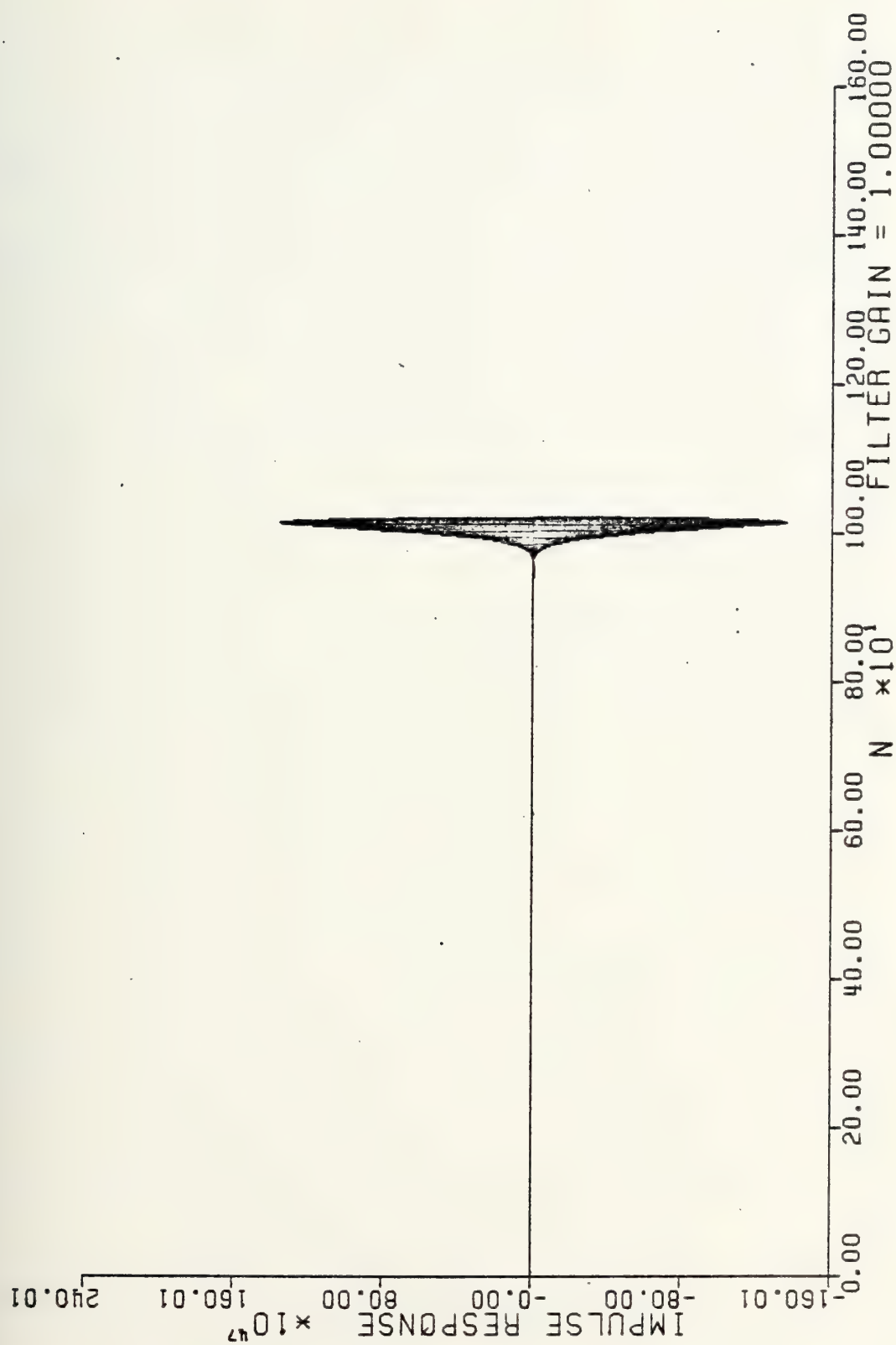


Figure 8-1b Unit Sample Response Of A Filter Exceeding System  
Dynamic Range Limitations



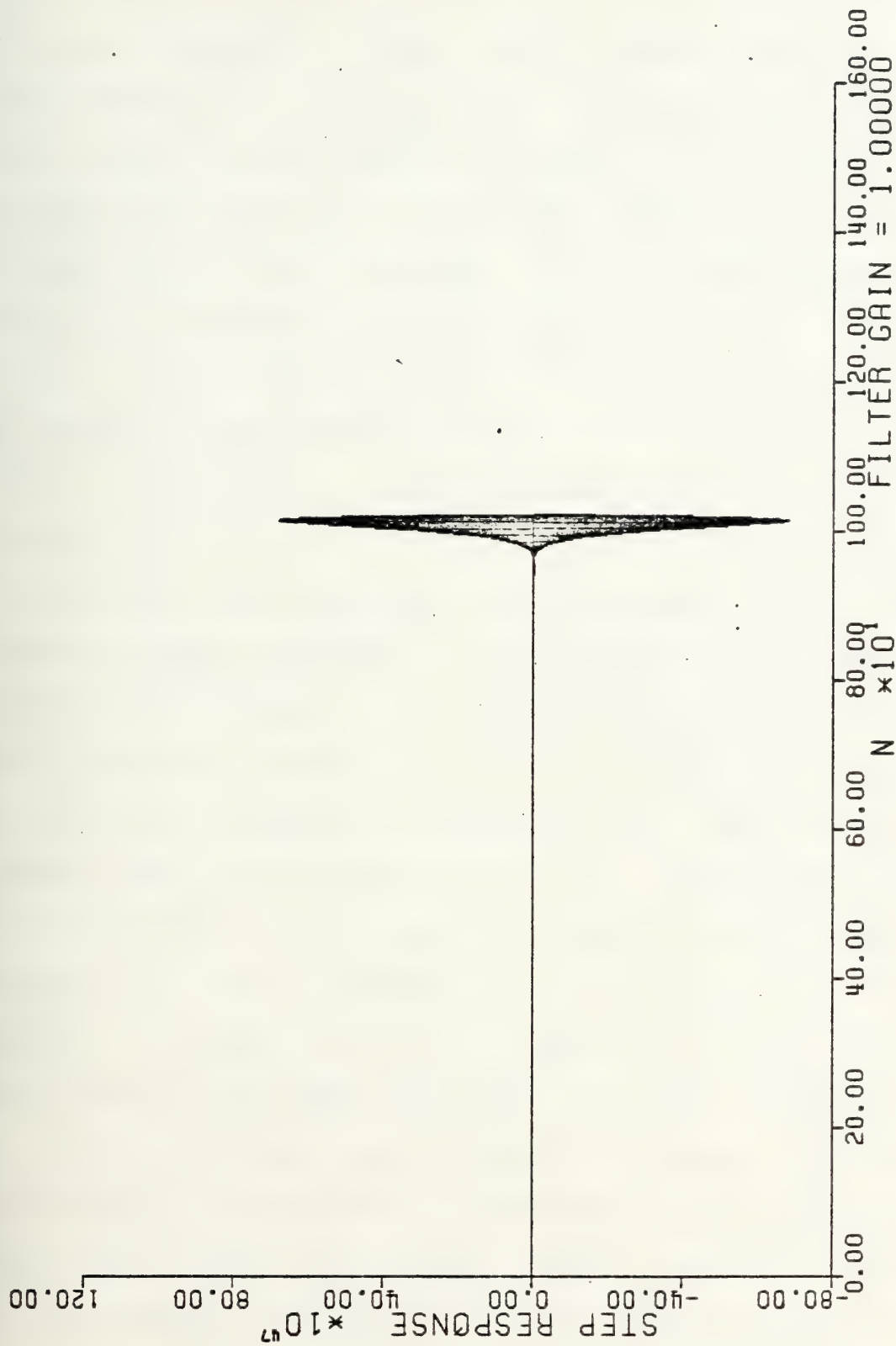


Figure 8-1c Step Response For A Filter Exceeding System Dynamic Range Limitations





larger of these limits is exceeded, the computations are terminated and it is not clear whether the designed filter is actually unstable or has merely exceeded the dynamic range limitations of the machine. The dedicated user may decide to modify the programs presented in the appendices to investigate the time responses of such systems. Clearly the form of the time responses will not be significantly altered if a modified unit sample response

$$1 \times 10^{-10}, 0, 0, 0, \dots$$

and a modified step response

$$1 \times 10^{-10}, 1 \times 10^{-10}, 1 \times 10^{-10}, \dots$$

is used.

If these modifications are performed, two other phenomena become apparent, each relating to the finite precision implementation of the digital filter. The first is that, in general, digital filters have deadbands, the second that the filter output may degenerate to a limit cycle. The original works by Blackman [Ref. 1] demonstrate that digital filters exhibit both of these effects. Extensive coverage of these phenomena is not appropriate here, however, a dead-band in a digital filter can be simplistically described as a span of filter input values for which the filter output does not change. The limit cycle effect is evident when a digital filter responds to a steady state input. For constant inputs, the filter output may run through a cycle of values. The propensity for



experiencing limit cycles is obvious in the calculation of the step response. A close look at the unit sample response shows that after a sufficiently large number of input values, this unit sample sequence becomes in reaching steady state, a constant.

When observing the output response of the example filter, there is still another possibility for explaining this output. This explanation revolves around the fact that the poles in the implemented filter are not at the locations expected. In fact, the poles may actually move outside the unit circle even though they were entered within. In general, there are only a finite number of pole locations possible with finite word length coefficient representations. The effect of this limitation is best explained using another example. Consider a filter with three zeros located at 0.5, and  $0.2 \pm j0.5$ , and four poles located at -0.999, -0.6, and  $-0.69 \pm j0.69$ . The infinite precision transfer function for this filter is expressed in equation 8.1.

$$\frac{z^3 - .9 z^2 + .49 z - .145}{z^4 + 2.979 z^3 + 3.75822 z^2 + 2.3497398 z + .57074868} \quad (8.1)$$

Now consider the finite precision case where each of the coefficients must be represented by a six bit number. The pole at -0.6 is expressed in binary (exclusive of sign) as



.10011001100. . . and is truncated to six bits as .100110. When all of the coefficients are so represented, the transfer function is expressed by equation 8.2. Clearly the poles and zeros of this second transfer function are not the

$$H(z) = \frac{z^3 - .890625 z^2 + .474275 z - .140625}{z^4 + 2.9375 z^3 + 3.75 z^2 + 2.3125 z + .5625} \quad (8.2)$$

same in equation 8.1. In general the differences will change all of the filter responses. In fact for this particular case the new poles of the system are at  $-0.70522 \pm j0.16677$ , and  $-0.76353 \pm j0.698616$ ; this filter is unstable!

The reason for the shift in pole locations is clearly portrayed by the filter whose characteristic equation is equation 8.3.

$$(z + .995)^8 = 0 \quad (8.3)$$

This filter has eight poles at  $-0.995$ . Note what happens to the filter when an error is introduced by truncating the infinite precision coefficient values to thirty-two bits. For simplicity restrict the discussion to the ramifications of an error generated in the value of the constant  $(.995)^8$ . This error can be as large as  $2 \times 10^{-10}$ . With this error explicitly included, the new characteristic equation will be:

$$(z + .995)^8 + 2 \times 10^{-10} = 0 \quad (8.4)$$



The roots of equation (8.4) are complex, and lie on a circle of radius 0.06 around the point  $-0.995 \pm j 0.0$ . Figure 8-2 shows that three of these roots lie outside the unit circle. Thus with a finite precision implementation of the filter (even with 32 bits of precision), it is possible for the pole locations to move so far that the filter actually becomes unstable.

The results of this section as well as four types of ASDF outputs which have been observed, are summarized in figure 8-3. Trace A shows the unit sample response of a stable filter. Trace B shows the response of a filter which exceeds the dynamic range limitations of the implementation system. Trace B1 is for an unstable filter and B2 is for a stable filter caught in a limit cycle. These two traces can be observed or explored only by modifying the unit sample sequence. Trace C shows a stable filter which falls into a limit cycle below the dynamic range limit of the machine upon which the implementation has been performed. Note that the performance of a filter above the dynamic range limitation of the IBM 360 (or any other machine) cannot be observed, unless the ASDF program is modified.







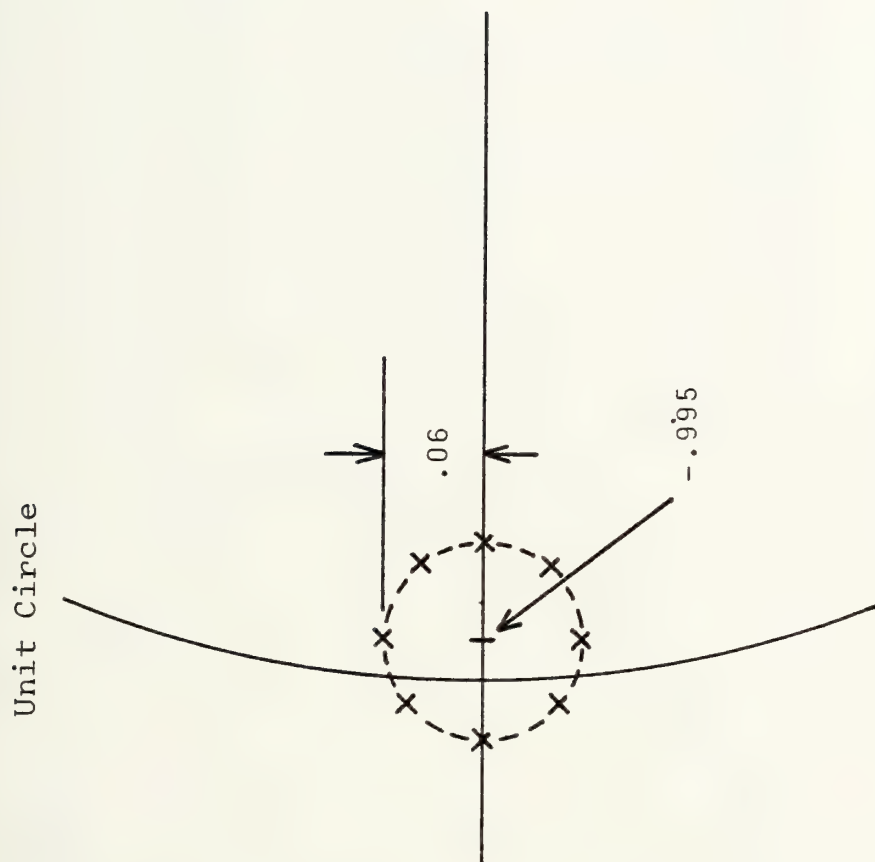


Figure 8-2 Three Of These Complex Poles Are Outside The Unit Circle



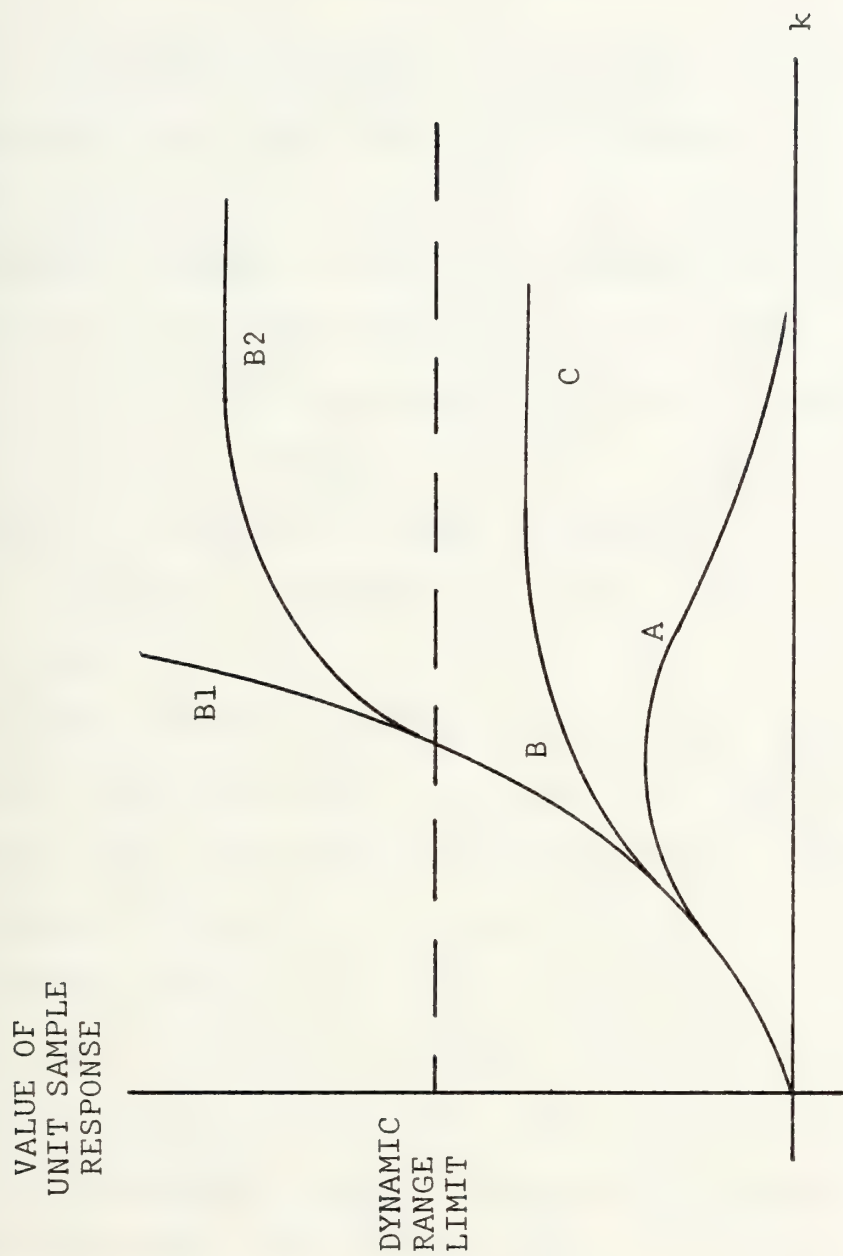


Figure 8-3 Summary Of Possible Observed Unit Sample Responses



## IX. SUMMARY AND CONCLUSIONS

The Advanced Simulation of Digital Filters has been implemented on the IBM 360, and is intended for the use by persons beginning their study of digital signal processing. The programs included within the package have been designed to make the user's interaction with the  $z$ -plane as simple as possible. While the programs themselves are available for study, all of the computations are invisible to the user, in an attempt to develop the user's intuition of the discrete domain without overwhelming him with arithmetic computations.

The ASDF package provides the user with an interactive method by which the filter pole and zero locations can be manipulated and the effects observed of such movement. ASDF can simulate filters of up to tenth order, and is punctuated with numerous comments and instructions which are to assist the user. The interface with the  $z$ -plane is available in a rapid interactive graphics mode, as well as in the slower polar and rectangular modes, each providing greater position location accuracy. The filter responses generated by ASDF are presented as a series of displays directly on the Tektronix 4012 terminal. Versatec plot capability has been integrated into the system to provide high quality copies of the system responses.



## A. CONCLUSIONS

There are several points which are worth considering for persons intending to pursue this or other areas of interactive systems development with the graphics terminals. While comments may appear obvious they are stated to minimize the difficulties which can be encountered. First, a serious attempt should be made to define the entire data structure before beginning the programming stage of the work. All phases of the effort must be planned before commencing with the actual programming to insure that the data structure selected is sufficiently general, and sufficiently flexible to endure the additions and changes which inevitably occur when initial objectives are overlooked or changed. Second, it is imperative to master all of the software tools available and applicable to the project. There are features of the PLOT 10 software which exist and were not used in the ASDF program. This shortcoming is due to insufficient a priori knowledge. For example, the writing of text portions of the screen presentations should have been much simpler. Finally, it is important to research all of the attributes of the devices used for the final problem solution, and direct the solution to take advantage of all the system capabilities.





## B. AREAS FOR FUTURE WORK

There are several areas in which this program could be improved or extended. The possibility exists for transferring the ASDF programs to the Tektronix 4082 computing system. This move would avoid many of the shortcomings associated with the current IBM 360 system. The user would be relatively immune to system crashes, and the constraints placed upon him by concerning the IBM system's being operational and/or available. There are several disadvantages associated with this move. With the advent of the new computer system the majority of the current problems concerning system reliability, and constraints on available access hours should be eliminated. Furthermore, transferring these programs to the 4082 system will require significant rewriting to make execution possible on a system which has a much smaller virtual machine. Furthermore to fully benefit from the 4082 system, the interfacing routines should be rewritten to take advantage of the 4082 refresh capability. It seems that all of the effort involved for this change is not worth the expected improvements.

The ASDF program is now useful for analyzing digital filters. It seems likely that this capability could be expanded into a method for computer-aided design of digital filters. There are standard transformations which allow the filter designer to map analog filter requirements into the



appropriate digital requirements, and as well there are transformations which make possible the computation of digital filter transfer functions from known analog transfer functions. There are also a variety of optimization techniques referenced in the literature which could be implemented.

The other area of obvious expansion for the ASDF program is to include the capability for performing spectral analysis. It would be worthwhile (though perhaps not particularly involved) to include in the system the ability to calculate Discrete Fourier Transforms, and graphically display the results. Additionally, it would be possible to demonstrate the correspondence between performing convolution in the time domain directly and performing the convolution in the frequency domain by multiplying the DFT's. The relation between the circular and linear convolution could also be graphically portrayed.

As a final precautionary note, it is not possible to overestimate the length of time required to generate well documented, correct, useful programs on an interactive basis. Changes in system parameters, and the limitations in system software can make the process of simplifying the user's interactive responsibilities a very difficult process for anyone not thoroughly acquainted with all available resources. The myriad of possible incorrect user responses which must be anticipated make the problem extensive.



## APPENDIX A: PROBABLE SOLUTIONS FOR DIFFICULTIES

SYMPTOM: RESPONSE command returns an error message upon execution.

SOLUTION: Check the virtual machine size with the command "CP Q F" to insure a virtual machine size of 400K. If the user has only been allocated 256K, logoff and log back on as directed in chapter five. Load ASDF as described in chapter six.

SYMPTOM: User has inadvertently entered the control program, ie., CP.

SOLUTION: Type in "b" and "return." Normal program execution should resume.

SYMPTOM: The carat appears with the blinking cursor, but it is not located in the command array.

SOLUTION: Move the trace cursor into the command array, and type a "space." Normal operation should resume.

SYMPTOM: No terminal activity.

SOLUTION: First, check to see whether or not the system has crashed. If the system has crashed, logon as in chapter five, and load ASDF as in chapter six. If the system has not crashed, move the cursor to the center of the screen and





type "space" and "return." If normal operation has not resumed hit the "break" key. When the CP prompt appears, type "b." Normal operation should resume.

SYMPTOM: An error is detected in the execution of ASDF191 EXEC indicating that no temporary disk space is available.

SOLUTION: Check with the consultant or a systems programmer.

SYMPTOM: The user can not escape from a delete pole or delete zero command in the interactive graphics mode (IAG).

SOLUTION: Implicit in each request to delete from the z-plane is a request, validating the change to be made. In the delete IAG mode the trace cursor returns after illustrating the root to be deleted. Type "y" if the correct root was identified. Type "n" if the wrong root was identified, and attempt the procedure again.

SYMPTOM: The user did not enter the correct amount of data and the program terminated on a 217 error.

SOLUTION: Type "b" and "return." The program should restart with the ASDF signature. At this point reenter the command desired. If executing the POLZRO command it is advisable to execute the list (LST) command to check on the current filter status.





## APPENDIX B: IMPORTANT SUBROUTINES AND VARIABLES

The subroutines and variables described in this appendix are applicable to all programs within the ASDF package.

### A. SUBROUTINE NAMES

The subroutines within the ASDF package are divided into three major categories: 1) Plot-10 software, 2) Versatec software, and 3) routines generated explicitly for the ASDF programs.

#### 1. Plot-10 Software Subroutines

The Plot-10 subroutines are all adequately described in reference 10, and are listed below.

|        |        |        |
|--------|--------|--------|
| ERASE  | MOVABS | ANMODE |
| RECOVR | HOME   | LINEF  |
| NEWLIN | SWINDO | VWINDO |
| POLTRN | MOVEA  | DRAWSA |
| LINTRN | DRAWR  | DRWREL |
| MOVREL | VCURSR | BELL   |
| POINTA | SCURSR | INIT   |
| TABHOR | TABVER | SEELOC |
| FIN    | FINITT | DRAWA  |
| DRWABS |        |        |



## 2. Versatec Subroutines

The Versatec subroutines are listed below, and are explained in detail in reference 7.

|       |      |        |
|-------|------|--------|
| PLOTS | PLOT | WINDOW |
| SCALE | AXIS | SYMBOL |
| LINE  |      |        |

## 3. ASDF Subroutines

Each of the following subroutines have been generated for the ASDF programs.

CRRMMM - Add and delete poles and zeros from the z-plane. The change "C" can be either "A" for add or "D" for delete. The roots "RR" can be "RZ" for real zero, "CZ" for complex zero, "RP" for real pole, or "CP" for complex pole. Mode "MMM" can be "IAG" for interactive graphics, "POL" for poles, or "RCT" for rectangular.

FLTR2 - Prints a listing of the pole and zero locations of the filter being considered.

ASUBK - Generates the  $a_k$  coefficients required to evaluate the transfer function.

BSUBR - Generates the  $b_r$  coefficients required to evaluate the transfer function.

ADJUST - Insures that the correct number of zeros located at the origin are included in the coefficients  $a_k$  and  $b_r$ .

RSPNSE - Computes the time responses of the filter.

FRQNCY - Computes the phase and magnitude of the transfer function for the prescribed filter.



PRPPLT - Scales the ordinate data prior to plotting the responses.

PLTIMP - Plots the unit sample response.

PLTSTP - Plots the unit step response.

PLTTRF - Plots the phase of the transfer function as well as the magnitude in linear and decibel format.

UNTCR - Plots the unit circle with the appropriate poles and zeros.

START - Initializes all arrays and required variables.

FINISH - Checks to insure that the constructed filter is causal, and generates a file to save the pole and zero locations.

ANGLE - Draws the angle sign for polar commands.

ERROR - Controls program flow and error messages for user generated errors.

LST - Prints a listing of the current pole zero location.

PLTTBL - Plots the poles and zeros on the unit circle.

CHCKSZ - Insures that roots are available to be deleted and that the maximum system order will not be exceeded.

UPDATE \_ Makes changes in the POLZRO array to correspond to user changes in the z-plane.

LOCATE - Searches the POLZRO table for the root closest to that which the user wishes to delete.

STRRTS - Puts poles and zeros into the POLZRO array.

PLOTRT - Plots a single pole or zero on the unit circle.



## B. IMPORTANT VARIABLES

The variables described below pertain to all of the subroutines and programs within the ASDF package.

IRROOTS(I) - Keeps track of the number of poles and zeros in the system under consideration. IRROOTS(1) is the number of real zeros. IRROOTS(2) is the number of complex zero pairs. IRROOTS(3) is the number of real poles. IRROOTS(4) is the number of complex pole pairs.

POLZRO(I,J) - The locations of the zeros are stored for values of I from one to ten. The pole locations are stored for values of I from 11 through 20. The real part of the rectangular representation of a root location is stored with J equal to one. The imaginary part of the rectangular representation for a root location is stored with J equal to two. The magnitude and angle of the root's polar representation are stored for J values of three and four, respectively. POLZRO(I,5) holds the root type.

IERROR - Keeps track of the generation of an error. Once the error has been identified the value is set to one. When the error has been explained to the user the value is reset to zero.

A(I) and B(I) - Store the values of the  $a_k$  and  $b_r$  coefficients.

AA(I) and BB(I) - Store the double precision values of the  $a_k$  and  $b_r$  coefficients.





YI(I) - Stores the values of the unit sample response to be plotted.

YS(I) - Stores the values of the unit step response to be plotted.

YP(I) - Stores the values of the phase of the transfer function to be plotted.

YM(I) - Stores the magnitudes of the transfer function to be plotted.

YMDB(I) - Stores the transfer function magnitudes in decibels to be plotted.

EN(I) - Stores the time values against which the dependent variables are plotted.

IYIPTS - Holds the number of unit sample points to be plotted.

IYSPTS - Holds the number of unit step response points to be plotted.

NDEN - Is the order of the denominator plus one.

NNUM - Is the order of the numerator plus one.

NORDER - Is the order of the system under consideration. This can be either of the numerator or denominator.

LAST - Holds the number of points to be plotted.

FLTRGN - Is the value that the user assigns to the filter gain.

ZEROS(I) - Holds the complex coefficients which will be made real to generate the  $b_r$  coefficients.



POLES(I) - Holds the complex coefficients which will be made real to generate the  $a_k$  coefficients.

The remainder of the variables are either obvious, or used merely for control of the program flow, ie. do loop indices, etc.



## APPENDIX C. STRUCTURE FOR ASDF TEXT FILE

The file ASDF TEXT is a composite file made up of all of the required programs and EXEC files for running the ASDF program package. These files have been consolidated to minimize the time required for transfer from DISK02 to OSX001. The structure of the composite file is listed below:

BREAK001

EXEC file ASDF192

BREAK002

TEXT file for POLZRO command

BREAK003

TEXT file for RESPONSE command

BREAK004

Information printed out by the INTRO command

BREAK005

VTEC1 JCL - used for EXEC HRD\$CPY

BREAK006

VTEC2 JCL - used for EXEC HRD\$CPY

BREAK007

Tables for RESPONSE plots

BREAK008

EXEC file to execute the EXEC PUNCH command

BREAK009



Initial file FT01F001 with no poles or zeros

BREAK010

TEXT file for generating JOB cards

BREAK011

TEXT file for the INTRO command

BREAK012

Information printed by OPTIONS command

BREAK013

Information printed on the thtle page

BREAK014

EXEC file to execute HRD\$CPY command

BREAK015

TEXT file for the OPTIONS command

BREAK016

TEXT file for the software page clear

BREAK017





## APPENDIX D. JCL FOR EXEC HRD\$CPY COMMAND

The Job Control Language required for the execution of the EXEC HRD\$CPY command is provided to give insight into the file manipulation utilized by the ASDF package. Should the permanent locations of any of the portions of the ASDF package change this section of JCL must be changed also.

### JCL for VTEC1

```
//GO EXEC PGM=THEPLT,REGION=180K
//STEPLIB DD UNIT=3330,VOL=SER=DISK03,DISP=SHR,DSN=S1502.ASDF
//FT05F001 DD DDNAME=SYSIN
//FT06F001 DD SYSOUT=A
//FT15F001 DD DDNAME=PLOTPARM
//PLOTLOG DD SYSOUT=A
//VECTR2 DD DSN=&&VECTR2,DISP=(,PASS),SPACE=(CYL,(5,5)),UNIT=SYSDA
//VECTR1 DD DSN=&&VECTR1,DISP=(,PASS),SPACE=(TRK,(1,1)),UNIT=SYSDA
//GO.SYSIN DD *
```

### JCL for VTEC2

```
//PLOT EXEC PGM=IEVMAPP,COND=(4,LT)
//FT15F001 DD DDNAME=PLOTPARM
//STEPLIB DD DSN=SYS1.VTECPLOT,DISP=SHR
//PLOTLOG DD SYSOUT=A
//VECTR1 DD DISP=(OLD,DELETE),DSN=&&VECTR1
//VECTR2 DD DISP=(OLD,DELETE),DSN=&&VECTR2
//SYSVECTR DD SYSOUT=(A,,5555)
//VECTTAPE DD DUMMY
//
```



AASDF1001  
AASDF1002  
AASDF1003  
AASDF1004  
AASDF1005  
AASDF1006  
AASDF1007  
AASDF1008  
AASDF1009  
AASDF1010  
AASDF1011  
AASDF1012  
AASDF1013  
AASDF1014  
AASDF1015  
AASDF1016  
AASDF1017  
AASDF1018  
AASDF1019  
AASDF1020  
AASDF1021  
AASDF1022  
AASDF1023  
AASDF1024  
AASDF1025  
AASDF1026  
AASDF1027  
AASDF1028  
AASDF1029  
AASDF1030  
AASDF1031  
AASDF1032  
AASDF1033  
AASDF1034  
AASDF1035  
AASDF1036  
AASDF1037  
AASDF1038  
AASDF1039  
AASDF1040  
AASDF1041  
AASDF1042  
AASDF1043  
AASDF1044  
AASDF1045  
AASDF1046  
AASDF1047  
AASDF1048

227







## APPENDIX F: SOURCE DECK FOR ASDFL92 EXEC

A SDF 2001  
A SDF 2002  
A SDF 2003  
A SDF 2004  
A SDF 2005  
A SDF 2006  
A SDF 2007  
A SDF 2008  
A SDF 2009  
A SDF 2010  
A SDF 2011  
A SDF 2012  
A SDF 2013  
A SDF 2014  
A SDF 2015  
A SDF 2016  
A SDF 2017  
A SDF 2018  
A SDF 2019  
A SDF 2020  
A SDF 2021  
A SDF 2022  
A SDF 2023  
A SDF 2024  
A SDF 2025  
A SDF 2026  
A SDF 2027  
A SDF 2028  
A SDF 2029  
A SDF 2030  
A SDF 2031  
A SDF 2032  
A SDF 2033  
A SDF 2034  
A SDF 2035  
A SDF 2036  
A SDF 2037  
A SDF 2038  
A SDF 2039  
A SDF 2040  
A SDF 2041  
A SDF 2042  
A SDF 2043  
A SDF 2044  
A SDF 2045  
A SDF 2046  
A SDF 2047  
A SDF 2048

[illegible]





[illegible]



ASDF2097  
ASDF2098  
ASDF2099  
ASDF2100  
ASDF2101  
ASDF2102  
ASDF2103  
ASDF2104  
ASDF2105  
ASDF2106  
ASDF2107  
ASDF2108  
ASDF2109  
ASDF2110  
ASDF2111  
ASDF2112  
ASDF2113  
ASDF2114  
ASDF2115  
ASDF2116  
ASDF2117  
ASDF2118  
ASDF2119  
ASDF2120  
ASDF2121  
ASDF2122  
ASDF2123  
ASDF2124  
ASDF2125  
ASDF2126  
ASDF2127  
ASDF2128  
ASDF2129  
ASDF2130  
ASDF2131  
ASDF2132  
ASDF2133  
ASDF2134  
ASDF2135  
ASDF2136  
ASDF2137  
ASDF2138  
ASDF2139  
ASDF2140  
ASDF2141  
ASDF2142  
ASDF2143  
ASDF2144

```

&COMMENT POLZRO TEXT P1
&EDIT RESPONSE TEXT P1
&EDIT INTRO FT04F001 P1
&EDIT VTEC1 JCL P1
&EDIT VTEC2 JCL P1
&EDIT FILE FT02F001 P1
&EDIT PUNCH EXEC P1
&EDIT FILE FT01F001 P1
&EDIT JOB$CRD TEXT P1
&EDIT INTRO TEXT P1
&EDIT FILE FT03F001 P1
&EDIT COVER FT04F001 P1
&EDIT HRD$CPY EXEC P1
&EDIT OPTIONS TEXT P1
&EDIT CLEAR TEXT P1
&COMMENT LOAD EACH OF THE CREATED FILES
&COMMENT AND GENERATE A CORE IMAGE
&COMMENT SO THAT SUBSEQUENT LOADING AND
&COMMENT LINKING IS NOT REQUIRED.
&COMMENT
LOAD POLZRO
GENMOD POLZRO (NOMAP P2)
LOAD RESPONSE
GENMOD RESPONSE (NOMAP P2)
LOAD INTRO
GENMOD INTRO
LOAD JOB$CRD
GENMOD JOB$CRD
LOAD OPTIONS
GENMOD OPTIONS (NOMAP P2)
LOAD CLEAR
GENMOD CLEAR (NOMAP P2)
&TYPEOUT OFF
ALTER COVER FT04F001 P1 FILE FT04F001 P1
INTRO
ALTER FILE FT04F001 P1 COVER FT04F001 P1
ALTER INTRO FT04F001 P1 FILE FT04F001 P1
OPTIONS
CLEAR
&COMMENT THESE NEXT STATEMENTS FORM AN
&COMMENT INFINITE LOOP WHICH PROCESSES
&COMMENT SUCCESSIVE ASDF AND CP/CMS COMMANDS.
-LOOP &PRINT *ASDF-READY
&ERROR &GOTO -LOOP2

```



CLEAR -LOOP  
&GOTO -LOOP  
-LOOP2 &PRINT INPUT COMMAND ERROR  
&GOTO -LOOP

ASDF2145  
ASDF2146  
ASDF2147  
ASDF2148





```

C*****
C* SOURCE DECK FOR ASDF COMMAND: POLZRO *
C*
C*****
C* THIS IS THE MAIN PROGRAM FROM WHICH ALL SUBROUTINES
C* WILL BE CALLED
C*
      COMMON IROOTS(4),POLZRO(20,5),IERROR,IJLD
      CALL START
      CALL UNICIR
      IF(IROOTS(1)+2*IROOTS(2).LE.IROOTS(3)+2*IROOTS(4))CALL FINISH
      IF(IROOTS(1)+2*IROOTS(2).LE.IROOTS(3)+2*IROOTS(4))STOP
      CALL ERASE
      CALL MOVABS(126,425)
      CALL ANMODE
      WRITE(6,100)
      FORMAT(,' FILTER IS NOT CAUSAL, TAKE APPROPRIATE CORRECTIVE',
100 X)
      CALL RECOVR
      CALL MOVABS(301,381)
      CALL ANMODE
      WRITE(6,102)
      FORMAT(,' HIT SPACE AND RETURN TO CONTINUE')
102
      CALL RECOVR
      CALL MOVABS(525,337)
      CALL ANMODE
      READ(5,101) IN
101
      FORMAT(A1)
      CALL RECOVR
      IOLD=1
      GO TO 10
      END

      SUBROUTINE LETTER
C* THIS SUBROUTINE GENERATES THE ALPHANUMERIC INFORMATION FOR
C* THE UNIT CIRCLE SUBROUTINE (32)
C*
      DIMENSION ICMJ(13)
      DATA ICMJ/3HACZ,3HACP,3HARZ,3HARP,
1 3HDCZ,3HDCP,3HDRZ,3HDRP,3HNUC,
2 3HLSI,3HEEM,3HRDY,3HEXC/
C* MOVE THE CURSOR TO THE SCREEN TOP LEFT CORNER (1)
C*
      CALL HOME

```





```

C* GENERATE A BLANK LINE (1)
C*
C* CALL LINEF
C* ANMODE IS REQUIRED FOR FORTRAN I/O IN GRAPHIC MODE (1)
C*
C* CALL ANMODE
C* PRINT THE TITLES (2)
C*
C* WRITE (6,100)
100 FORMAT (6X,20HUNIT CIRCLE COMMANDS)
C* GET THE TERMINAL BACK TO THE GRAPHIC MODE (1)
C*
C* CALL RECOVR
C* CALL NEWLIN
C* CALL LINEF
C* CALL ANMODE
C* WRITE (6,101)
101 FORMAT (9X,18HINTER- KEYBOARD)
C* CALL RECOVR
C* GENERATE A CARRIAGE RETURN AND A LINE FEED (1)
C*
C* CALL NEWLIN
C* CALL ANMODE
C* WRITE (6,102)
102 FORMAT (3X,25HCMDS ACTIVE RECT POLAR)
C* CALL RECOVR
C* CALL NEWLIN
C* CALL LINEF
C* PRINT THE COMMANDS (6)
C*
C* DO 40 M=1,10
C* CALL ANMODE
C* WRITE (6,103) ICMD(M)
C* CALL RECOVR
C* CALL LINEF
C* FORMAT (4X,A3)
40 RETURN
103 END
C* SUBROUTINE PLOTUC
C* THIS SUBROUTINE DRAWS THE UNIT CIRCLE WITH THE AXES

```

ASDF3049  
ASDF3050  
ASDF3051  
ASDF3052  
ASDF3053  
ASDF3054  
ASDF3055  
ASDF3056  
ASDF3057  
ASDF3058  
ASDF3059  
ASDF3060  
ASDF3061  
ASDF3062  
ASDF3063  
ASDF3064  
ASDF3065  
ASDF3066  
ASDF3067  
ASDF3068  
ASDF3069  
ASDF3070  
ASDF3071  
ASDF3072  
ASDF3073  
ASDF3074  
ASDF3075  
ASDF3076  
ASDF3077  
ASDF3078  
ASDF3079  
ASDF3080  
ASDF3081  
ASDF3082  
ASDF3083  
ASDF3084  
ASDF3085  
ASDF3086  
ASDF3087  
ASDF3088  
ASDF3089  
ASDF3090  
ASDF3091  
ASDF3092  
ASDF3093  
ASDF3094  
ASDF3095  
ASDF3096



```

C* CONFINE THE UNIT CIRCLE TO A 600 X 600 RASTER UNIT SQUARE
C* POSITION THE SCREEN WINDOW
C* XMIN = (1023-600) = 423
C* YMIN = 0
C*
C* / CALL SWINDO (393,630,0,630)
C*
C* GENERATE THE UNIT CIRCLE (4)
C*
C* CALL VWINDO (0.,1.,1,0.,360.)
C* CALL POLTRN (0.,360.,0.)
C* CALL MOVEA (1.,0.)
C* CALL DRAWSA (1.,360.)
C*
C* RETURN TO RECTANGULAR COORDINATES
C* VIRTUAL WINDOW: -1.1 LE X,Y LE +1.1 (2)
C*
C* CALL LINTRN
C* CALL VWINDO (-1.1,2.2,-1.1,2.2)
C*
C* GENERATE THE X-AXIS (2)
C*
C* CALL MOVEA (1.,0.)
C* CALL DRAWR (-2.,0.)
C*
C* GENERATE THE Y-AXIS (2)
C*
C* CALL MOVEA (0.,1.)
C* CALL DRAWR (0.,-2.)
C*
C* GENERATE THE X-AXIS TICK MARKS (5)
C* B IS THE TICK MARK LENGTH IN USER UNITS
C*
C* DO 10 I=1,19
C* B=.05
C* IF(I.EQ.5.OR.I.EQ.15)B=.1
C* CALL MOVEA (-1.,+FLOAT(I)*.1,-B/2.)
C* CALL DRAWR (0.,B)
C* 10
C*
C* GENERATE THE Y-AXIS TICK MARKS (5)
C* B IS THE TICK MARK LENGTH IN USER UNITS
C*
C* DO 20 I=1,19
C* B=.05
C* IF(I.EQ.5.OR.I.EQ.15)B=.1
C* CALL MOVEA (-B/2.,1.-FLOAT(I)*.1)
C* CALL DRAWR (B,0.)
C* 20

```

ASDF3097  
ASDF3098  
ASDF3099  
ASDF3100  
ASDF3101  
ASDF3102  
ASDF3103  
ASDF3104  
ASDF3105  
ASDF3106  
ASDF3107  
ASDF3108  
ASDF3109  
ASDF3110  
ASDF3111  
ASDF3112  
ASDF3113  
ASDF3114  
ASDF3115  
ASDF3116  
ASDF3117  
ASDF3118  
ASDF3119  
ASDF3120  
ASDF3121  
ASDF3122  
ASDF3123  
ASDF3124  
ASDF3125  
ASDF3126  
ASDF3127  
ASDF3128  
ASDF3129  
ASDF3130  
ASDF3131  
ASDF3132  
ASDF3133  
ASDF3134  
ASDF3135  
ASDF3136  
ASDF3137  
ASDF3138  
ASDF3139  
ASDF3140  
ASDF3141  
ASDF3142  
ASDF3143  
ASDF3144



RETURN  
END

# SUBROUTINE BOXUC

C\* THE PURPOSE OF THIS SUBROUTINE IS TO GENERATE THE "CHOICE"  
C\* BOXES FOR THE UNIT CIRCLE SUBROUTINE  
C\*

C\* DIMENSION IBCORD (3,3)

C\* DATA DESCRIBING THE LOWER LEFT CORNER (LLC) OF  
C\* EACH BOX AND THE NUMBER OF BOXES IN EACH COLUMN  
C\* IBCORD (J,1) IBCORD (J,2) IBCORD (J,3)  
C\* LLCX LLCY NO. BOXES  
C\*

C\* DATA IBCORD/140,238,336,439,483,483,10,8,8/  
C\*

C\* GENERATE THE "CHOICE" BOXES (10)  
C\*

C\* DO 10 J=1,3  
C\*

C\* MOVE TO THE LOWER LEFT CORNER OF THE BOX (1)  
C\*

C\* CALL MOVABS (IBCORD(J,1),IBCORD(J,2))  
C\*

C\* "IBCORD(J,3)" CONTROLS THE NUMBER OF BLOCKS TO BE DRAWN (10)  
C\*

C\* K=IBCORD(J,3)  
C\* DO 20 I=1,K  
C\*

C\* DRAW THE ACTUAL BOX (4)  
C\*

C\* CALL DRWREL (0,16)  
C\* CALL DRWREL (14,0)  
C\* CALL DRWREL (0,-16)  
C\* CALL DRWREL (-14,0)  
C\*

C\* PROCEED TO THE NEXT HIGHER BOX IN THE COLUMN (1)  
C\*

C\* 20 CALL MOVREL (0,22)  
C\* 10 CONTINUE  
C\* RETURN  
C\* END

SUBROUTINE ACZIAG  
COMMON IROOTS(4),POLZRO(20,5), IERROR  
CALL CHCKSZ (2,2)  
IF(IERROR.NE.0)RETURN

ASDF3145  
ASDF3146  
ASDF3147  
ASDF3148  
ASDF3149  
ASDF3150  
ASDF3151  
ASDF3152  
ASDF3153  
ASDF3154  
ASDF3155  
ASDF3156  
ASDF3157  
ASDF3158  
ASDF3159  
ASDF3160  
ASDF3161  
ASDF3162  
ASDF3163  
ASDF3164  
ASDF3165  
ASDF3166  
ASDF3167  
ASDF3168  
ASDF3169  
ASDF3170  
ASDF3171  
ASDF3172  
ASDF3173  
ASDF3174  
ASDF3175  
ASDF3176  
ASDF3177  
ASDF3178  
ASDF3179  
ASDF3180  
ASDF3181  
ASDF3182  
ASDF3183  
ASDF3184  
ASDF3185  
ASDF3186  
ASDF3187  
ASDF3188  
ASDF3189  
ASDF3190  
ASDF3191  
ASDF3192





```

CALL VCURSR (ICCHAR,X,Y)
CALL STRRTS (2,1,X,Y)
RETURN
END

```

```

SUBROUTINE ACPIAG
COMMON IXXXXX(4), PXXXXX(20,5), IERROR
CALL CHCKSZ (2,4)
IF(IERROR.NE.0)RETURN
CALL VCURSR (ICCHAR, X,Y)

```

```

C* INSURE THAT THE POLES ARE WITHIN THE UNIT CIRCLE
C*
C*

```

```

IF(SQRT(X**2+Y**2).GT.1)IERROR=9
IF(IERROR.NE.0)RETURN
CALL STRRTS (4,1,X,Y)
RETURN
END

```

```

SUBROUTINE ARZIAG
COMMON IXXXXX(4), PXXXXX(20,5), IERROR
CALL CHCKSZ(1,1)
IF(IERROR.NE.0)RETURN
CALL VCURSR(ICCHAR,X,Y)
Y = 0
CALL STRRTS (1,1,X,Y)
RETURN
END

```

```

SUBROUTINE ARPIAG
COMMON IXXXXX(4), PXXXXX(20,5), IERROR
CALL CHCKSZ (1,3)
IF(IERROR.NE.0)RETURN
CALL VCURSR (ICCHAR, X, Y)

```

```

C* INSURE THAT THE POLE IS WITHIN THE UNIT CIRCLE
C*
C*

```

```

Y=0
IF(SQRT(X**2+Y**2).GT.1)IERROR=9
IF(IERROR.NE.0)RETURN
CALL STRRTS (3,1,X,Y)
RETURN
END

```

```

SUBROUTINE DCZIAG
COMMON IXXXXX(4), POLZRO(20,5), IERROR
CALL CHCKSZ(-1,2)
IF(IERROR.NE.0)RETURN

```

```

ASDF3193
ASDF3194
ASDF3195
ASDF3196
ASDF3197
ASDF3198
ASDF3199
ASDF3200
ASDF3201
ASDF3202
ASDF3203
ASDF3204
ASDF3205
ASDF3206
ASDF3207
ASDF3208
ASDF3209
ASDF3210
ASDF3211
ASDF3212
ASDF3213
ASDF3214
ASDF3215
ASDF3216
ASDF3217
ASDF3218
ASDF3219
ASDF3220
ASDF3221
ASDF3222
ASDF3223
ASDF3224
ASDF3225
ASDF3226
ASDF3227
ASDF3228
ASDF3229
ASDF3230
ASDF3231
ASDF3232
ASDF3233
ASDF3234
ASDF3235
ASDF3236
ASDF3237
ASDF3238
ASDF3239
ASDF3240

```





```

10      CALL VCURSR(ICHAR,X,Y)
C*      CALL LOCATE(2,X,Y,LINEPZ)
C*      INDICATE THE COMPLEX ZERO NEAREST THE USER'S CURSOR
C*      WITH A BRIGHT DOT FIVE TIMES (4)
C*
      XX=POLZRO(LINEPZ,1)
      YY=POLZRO(LINEPZ,2)
      IF(POLZRO(LINEPZ,3).LE.1.0)GO TO 15
      ANGLE=ATAN2(YY,XX)
      XX=1.05*COS(ANGLE)
      YY=1.05*SIN(ANGLE)
      DO 20 NREPS=1,5
15      CALL BELL
      IF(Y.LE.0.)CALL POINTA(XX,-YY)
      IF(Y.GT.0.)CALL POINTA(XX,YY)
      CALL SCURSR(ICHAR,IX,IY)
20      IF(ICHAR.NE.121.AND.ICHAR.NE.89)GO TO 10
      CALL UPDATE(2,LINEPZ)
      CALL NUC
      RETURN
      END

      SUBROUTINE DCPIAG
      COMMON IXXXXX(4),POLZRO(20,5),IERROR
      CALL CHCKSZ(-1,4)
      IF(IERROR.NE.0)RETURN
      CALL VCURSR (ICHAR,X,Y)
      CALL LOCATE(4,X,Y,LINEPZ)
10
C*      INDICATE THE COMPLEX POLE NEAREST THE USER'S CURSOR
C*      FIVE TIMES WITH A BRIGHT DOT (4)
C*
      DO 20 NREPS=1,5
      CALL BELL
      IF(Y.LE.0.)CALL POINTA(POLZRO(LINEPZ,1),-POLZRO(LINEPZ,2))
      IF(Y.GT.0.)CALL POINTA(POLZRO(LINEPZ,1),POLZRO(LINEPZ,2))
20      CALL SCURSR (ICHAR,IX,IY)
      IF(ICHAR.NE.121.AND.ICHAR.NE.89)GO TO 10
      CALL UPDATE (4,LINEPZ)
      CALL NUC
      RETURN
      END

      SUBROUTINE DRZIAG
      COMMON IXXXXX(4),POLZRO(20,5),IERROR
      CALL CHCKSZ (-1,1)
      IF(IERROR.NE.0)RETURN

```



```

10 CALL VCURSR(ICHAR,X,Y)
   CALL LOCATE(1,X,0.,LINEPZ)
   DO 20 NREPS=1,5
20  CALL BELL
   CALL POINTA(POLZRO(LINEPZ,1),0.)
   CALL SCURSR (ICHAR,IX,IY)
   IF(ICHAR.NE.121.AND.ICHAR.NE.89)GO TO 10
   CALL UPDATE(1,LINEPZ)
   CALL NUC
   RETURN
   END

SUBROUTINE DRPIAG
COMMON IXXXX(4),POLZRO(20,5),IERROR
CALL CHCKSZ(-1,3)
IF(IERROR.NE.0)RETURN
CALL VCURSR(ICHAR,X,Y)
CALL LOCATE(3,X,0.,LINEPZ)
DO 20 NREPS=1,5
20  CALL BELL
   CALL POINTA(POLZRO(LINEPZ,1),0.)
   CALL SCURSR(ICHAR,IX,IY)
   IF(ICHAR.NE.121.AND.ICHAR.NE.89)GO TO 10
   CALL UPDATE(3,LINEPZ)
   CALL NUC
   RETURN
   END

SUBROUTINE UNTCIR
C* THIS SUBROUTINE CONTROLS THE IMPLEMENTATION OF
C* COMMANDS ASSOCIATED WITH THE UNIT CIRCLE
C*
DATA ISE/2H E/
DATA IX/2H X/
DIMENSION IBCORD(4,3),IHIBL(10),IVIBL(10)
COMMON IXXXX(4),PXXXX(20,5),IERROR,IOLD
C* SET THE TAB TABLE SIZE TO TEN
C*
C* CALL TTBLSZ(10)
C* LOCATE THE HORIZONTAL AND VERTICAL TABS (2)
C*
DATA IHIBL/128,226,324,0,0,0,0,0,0,0,0,0/
DATA IVIBL/439,461,483,505,527,549,571,593,615,637/
C* INITIATE THE GRAPHICS ROUTINES (1)
C*

```



```

C*      IF(IOLD.EQ.0)CALL INIT
C*
C*      IF WORKING ON AN OLD FILTER - CONTINUE WITH IT
C*
C*      IF(IOLD.EQ.1)CALL NUC
C*      IF(IOLD.EQ.1)GO TO 30
C*      IF(IOLD.EQ.0)CALL OLFLTR
C*      IF(IOLD.EQ.0)CALL ERASE
C*      IF(IOLD.EQ.1)CALL NUC
C*      IF(IOLD.EQ.1)GO TO 30
C*
C*      GENERATE THE ALPHANUMERIC I/O FOR THE COMMAND TABLE
C*      OF THE UNIT CIRCLE SUBROUTINE (1)
C*
C*      CALL LETTER
C*
C*      GENERATE THE "CHOICE" BOXES (1)
C*
C*      CALL BOXUC
C*
C*      GENERATE THE UNIT CIRCLE (1)
C*
C*      CALL PLOTUC
C*
C*      ALLOW THE USER TO SELECT THE DESIRED COMMAND (1)
C*
C*      CALL BELL
C*      CALL SCURSR (IIS,IIX,IYY)
C*
C*      ADJUST CURSOR FOR INACCURACY ON USER'S PART (1)
C*
C*      CALL MOVABS (IIX-63,IYY+3)
C*
C*      PUT THE ALPHANUMERIC CURSOR IN THE NEAREST BOX (2)
C*
C*      CALL TABHOR(IHTBL)
C*      CALL TABVER(IVTBL)
C*
C*      RECORD THE LOCATION OF THE CURSOR AFTER
C*      THE TAB COMMAND HAS BEEN EXECUTED (1)
C*
C*      CALL SEELC (IA,IB)
C*
C*      HIGHLIGHT THE CHOSEN BLOCK / COMMAND WITH THE FLASHING CURSOR (4)
C*
C*      CALL ANMODE
C*      READ (5,104) IZ

```





```

104  FORMAT(A2)
      CALL RECOVER
C*  IF HE HAS RESPONDED WITH "SPACE" AND "X"
C*  TERMINATE PROGRAM EXECUTION (2)
C*
C*  IF (IZ.EQ.IX) RETURN
C*
C*  IF THE USER HAS NOT RESPONDED WITH A "SPACE"
C*  AND AN "E" THEN DO NOT EXECUTE THIS COMMAND
C*  RETURN HIM TO THE CURSOR (1)
C*
C*  IF (IZ.NE.ISE) GO TO 30
C*
C*  ROUTE THE PROGRAM CONTROL TO THE APPROPRIATE (34)
C*  SUBROUTINE BASED ON THE TABBED CURSOR LOCATION
C*
C*  40
      IF (IA.EQ.128.AND.IB.EQ.637) CALL ACZIAG
      IF (IA.EQ.128.AND.IB.EQ.615) CALL ACPIAG
      IF (IA.EQ.128.AND.IB.EQ.593) CALL ARZIAG
      IF (IA.EQ.128.AND.IB.EQ.571) CALL ARPIAG
      IF (IA.EQ.128.AND.IB.EQ.549) CALL DCZIAG
      IF (IA.EQ.128.AND.IB.EQ.527) CALL DCPIAG
      IF (IA.EQ.128.AND.IB.EQ.505) CALL DRZIAG
      IF (IA.EQ.128.AND.IB.EQ.483) CALL DRPIAG
      IF (IA.EQ.128.AND.IB.EQ.461) CALL NUC
      IF (IA.EQ.128.AND.IB.EQ.439) CALL LST
      IF (IA.EQ.226.AND.IB.EQ.637) CALL ACZRCT
      IF (IA.EQ.226.AND.IB.EQ.615) CALL ACPRCT
      IF (IA.EQ.226.AND.IB.EQ.593) CALL ARZRCT
      IF (IA.EQ.226.AND.IB.EQ.571) CALL ARPRCT
      IF (IA.EQ.226.AND.IB.EQ.549) CALL DCZRCT
      IF (IA.EQ.226.AND.IB.EQ.527) CALL DCPRCT
      IF (IA.EQ.226.AND.IB.EQ.505) CALL DRZRCT
      IF (IA.EQ.324.AND.IB.EQ.483) CALL DRPRCT
      IF (IA.EQ.324.AND.IB.EQ.461) CALL ACZPLR
      IF (IA.EQ.324.AND.IB.EQ.439) CALL ACPLR
      IF (IA.EQ.324.AND.IB.EQ.417) CALL ARZPLR
      IF (IA.EQ.324.AND.IB.EQ.395) CALL ARPLR
      IF (IA.EQ.324.AND.IB.EQ.373) CALL DCZPLR
      IF (IA.EQ.324.AND.IB.EQ.351) CALL DCPLR
      IF (IA.EQ.324.AND.IB.EQ.329) CALL DRZPLR
      IF (IA.EQ.324.AND.IB.EQ.307) CALL DRPLR
      IF (IERROR.NE.0) CALL ERROR
      GO TO 30
END
SUBROUTINE OLFLTR

```





```

COMMON IROOTS(4),POLZRO(20,5),IERROR,IOLD,FLTRGN
CALL MOVABS(126,374)
CALL ANMODE
WRITE(6,100)
FORMAT(,' DO YOU HAVE AN OLD FILTER THAT YOU WISH TO MODIFY?')
CALL RECOVER
CALL MOVABS(399,330)
CALL ANMODE
WRITE(6,101)
FORMAT(,' TYPE Y OR N')
CALL RECOVER
CALL SCURSR(IN,IX,IY)
IF(IN.NE.121.AND.IN.NE.89)RETURN
DO 10 I=1,20
  READ(1,104) (POLZRO(I,J),J=1,5)
  FORMAT(1X,5F12.8)
  READ(1,105) (IROOTS(I),I=1,4)
  FORMAT(1X,4I2)
  READ(1,106) FLTRGN
  FORMAT(1X,F16.8)
  IOLD=I
  RETURN
END

SUBROUTINE FINISH
COMMON IROOTS(4),POLZRO(20,5),IERROR,IOLD,FLTRGN
CALL ERASE
CALL ANMODE
REWIND 1
DO 10 I=1,20
  WRITE(1,100) (POLZRO(I,J), J=1,5)
  FORMAT(1X,5F12.8)
  CONTINUE
  WRITE(1,101) (IROOTS(I),I=1,4)
  FORMAT(1X,4I2)
  IF(IOLD.EQ.0)FLTRGN=1.
  CALL RECOVER
  CALL MOVABS(217,528)
  CALL ANMODE
  WRITE(6,102) FLTRGN
  FORMAT(,' THE FILTER GAIN IS CURRENTLY: ',F12.8)
  CALL RECOVER
  CALL MOVABS(252,484)
  CALL ANMODE
  WRITE(6,103)
  FORMAT(,' DO YOU WISH TO CHANGE THE FILTER GAIN?')
  CALL RECOVER
  CALL MOVABS(427,440)

```

```

ASDF3433
ASDF3434
ASDF3435
ASDF3436
ASDF3437
ASDF3438
ASDF3439
ASDF3440
ASDF3441
ASDF3442
ASDF3443
ASDF3444
ASDF3445
ASDF3446
ASDF3447
ASDF3448
ASDF3449
ASDF3450
ASDF3451
ASDF3452
ASDF3453
ASDF3454
ASDF3455
ASDF3456
ASDF3457
ASDF3458
ASDF3459
ASDF3460
ASDF3461
ASDF3462
ASDF3463
ASDF3464
ASDF3465
ASDF3466
ASDF3467
ASDF3468
ASDF3469
ASDF3470
ASDF3471
ASDF3472
ASDF3473
ASDF3474
ASDF3475
ASDF3476
ASDF3477
ASDF3478
ASDF3479
ASDF3480

```



```

104 CALL ANMODE
WRITE(6,104)
FORMAT(1,(TYPE Y OR N)*)
CALL RECOVR (NEWGN,IX,IY)
CALL SCURSR (NEWGN,IX,IY)
CALL ANMODE
IF(NEWGN.NE.121.AND.NEWGN.NE.89)GO TO 30
CALL RECOVR
CALL MOVABS (182,396)
CALL ANMODE
WRITE(6,106)
FORMAT(1,TYPE IN THE NEW FILTER GAIN, 12 OR FEWER NUMBERS*)
CALL RECOVR
CALL MOVABS (357,374)
CALL ANMODE
WRITE(6,109)
FORMAT(1,INCLUDE A DECIMAL POINT*)
CALL RECOVR
CALL MOVABS (421,330)
CALL ANMODE
READ(5,107) FLTRGN
FORMAT(1,F12.8)
CALL RECOVR
CALL MOVABS (259,286)
CALL ANMODE
WRITE(6,108) FLTRGN
FORMAT(1,NEW FILTER GAIN IS: ',F16.8)
FORMAT(1X,F16.8)
CALL ANMODE
WRITE(1,110) FLTRGN
CALL FIN
RETURN
END

SUBROUTINE NUC
CALL ERASE
CALL LETTER
CALL BOXUC
CALL PLOTUC
CALL PLTTBL
RETURN
END

SUBROUTINE ACZRC
COMMON IXXXX(4),POLZRO(20,5),IERROR
DIMENSION LTRACZ(64)
DATA LTRACZ/4H0 A,4HDD A,4H COM,4HPLEX,4H ZER,4H0 TO,4H THE
X ,4H ,4HDI SP,4HLAY,,4H ENT,4HER T,4HHE R,4HEAL ,4HAND

```

ASDF3481  
ASDF3482  
ASDF3483  
ASDF3484  
ASDF3485  
ASDF3486  
ASDF3487  
ASDF3488  
ASDF3489  
ASDF3490  
ASDF3491  
ASDF3492  
ASDF3493  
ASDF3494  
ASDF3495  
ASDF3496  
ASDF3497  
ASDF3498  
ASDF3499  
ASDF3500  
ASDF3501  
ASDF3502  
ASDF3503  
ASDF3504  
ASDF3505  
ASDF3506  
ASDF3507  
ASDF3508  
ASDF3509  
ASDF3510  
ASDF3511  
ASDF3512  
ASDF3513  
ASDF3514  
ASDF3515  
ASDF3516  
ASDF3517  
ASDF3518  
ASDF3519  
ASDF3520  
ASDF3521  
ASDF3522  
ASDF3523  
ASDF3524  
ASDF3525  
ASDF3526  
ASDF3527  
ASDF3528



```

X      ,4HIMAG,4HINAR,4HY PA,4HRTS ,4HWITH,4HIN T,4HHE B
X      ,4HBOXES,4HPROV,4HIDED,4H. EA,4HCH N,4HUMBE,4HR RE,4HQURI
X      ,4HES A,4HDECI,4HMAL.,4H REA,4HL PA,4HRT M,4HINUS,4H SIG
X      ,4HNS ,4HMUST,4H BE ,4HINCL,4HUDED,4H WIT,4HHIN ,4HTHE
X      ,4HBOX.,4HNOTE,4H: AL,4HL CO,4HMPLE,4HX ZE,4HROES,4H MUS
X      ,4HT BE,4HWITH,4HIN T,4HEN U,4HNITS,4H OF ,4HTHE ,4HORIG/
X      DATA LTRACZ(64)/4HIN. /
10    CALL CHCKSZ(2,2)
      IF(IEERROR.NE.0) RETURN
      CALL ERASE (0,528)
      DO 30 I=1,57,8
      CALL ANMODE
      WRITE(6,100) (LTRACZ(I+J-1),J=1,8)
30    CALL RECOVR
100   CALL NEWLIN (21X,8A4)
      CALL NEWLIN
      CALL ANMODE
      WRITE (6,101)
101   FORMAT (27X,4HREAL,9X,9HIMAGINARY)
      CALL RECOVR
      CALL NEWLIN
      CALL ANMODE
      WRITE (6,102)
102   FORMAT(34X,5H+/- J)
      CALL RECOVR
      CALL MOVABS (320,302)
      CALL DRWREL(0,22)
      CALL DRWREL(138,0)
      CALL DRWREL(0,-22)
      CALL DRWREL(-138,0)
      CALL DRWREL(212,0)
      CALL DRWREL(0,22)
      CALL DRWREL(140,0)
      CALL DRWREL(0,-22)
      CALL DRWREL(-140,0)
      CALL MOVREL (0,3)
      CALL CARTN
      CALL ANMODE
      WRITE (6,103)
103   FORMAT (22X,'>')
      READ (5,104) X,Y
104   FORMAT(F10.7,5X,F10.7)
      CALL RECOVR
      IF(SQRT(X**2+Y**2).GE.10.) IEERROR=10
      IF(IEERROR.NE.0) RETURN
      CALL NEWLIN

```

```

ASDF3529
ASDF3530
ASDF3531
ASDF3532
ASDF3533
ASDF3534
ASDF3535
ASDF3536
ASDF3537
ASDF3538
ASDF3539
ASDF3540
ASDF3541
ASDF3542
ASDF3543
ASDF3544
ASDF3545
ASDF3546
ASDF3547
ASDF3548
ASDF3549
ASDF3550
ASDF3551
ASDF3552
ASDF3553
ASDF3554
ASDF3555
ASDF3556
ASDF3557
ASDF3558
ASDF3559
ASDF3560
ASDF3561
ASDF3562
ASDF3563
ASDF3564
ASDF3565
ASDF3566
ASDF3567
ASDF3568
ASDF3569
ASDF3570
ASDF3571
ASDF3572
ASDF3573
ASDF3574
ASDF3575
ASDF3576

```











```

102  FORMAT (34X,5H+/- J)
      CALL RECOVER (320,302)
      CALL MOVABS (0,22)
      CALL DRWREL (138,0)
      CALL DRWREL (0,-22)
      CALL MOVREL (-138,0)
      CALL DRWREL (212,0)
      CALL DRWREL (0,22)
      CALL DRWREL (140,0)
      CALL DRWREL (0,-22)
      CALL MOVREL (-140,0)
      CALL ANMODE (0,3)
      CALL CARTN
      CALL ANMODE
      WRITE (6,103)
103  FORMAT (22X,'>')
      READ (5,104) X,Y
104  FORMAT (F10.7,5X,F10.7)
      CALL RECOVER (X**2+Y**2).GT.1.0) IERROR=10
      IF (IERROR.NE.0) RETURN
      CALL NEWLIN
      CALL ANMODE
      Z=ABS(Y)
      WRITE (6,105) X,Z
105  FORMAT (9X,29H THE NEW COMPLEX POLE WILL BE ,F10.7,6H +/- J,F10.7)
      CALL RECOVER
      CALL NEWLIN
      CALL ANMODE
      WRITE (6,106)
106  FORMAT (21X,32H IF CORRECT TYPE 'Y', IF NOT 'N'.)
      CALL RECOVER
      CALL SCURSR (ICAR,IX,IY)
      IF (ICAR.NE.121.AND. ICAR.NE.89) GO TO 10
      CALL NUC
      CALL STRTS (4,1,X,Y)
      RETURN
      END

SUBROUTINE ARZCT
COMMON IXXXX(4),POLZRO(20,5),IERROR
DIMENSION LTRARZ(64)
DATA LTRARZ/4HTO A,4HDD A,4H REA,4HL ZE,4HRO T,4HQ TH,4HE DI
      ,4HS- ,4HPLAY,4H ENT,4HER T,4HHE R,4HEAL ,4HPART,4H WIT
      ,4HHIN ,4HTHE ,4HBOX ,4HPROV,4HIDE,4H. TH,4HE NU,4HMBER
      ,4H IS ,4HTO I,4HNCLU,4HDE A,4H DEC,4HIMAL,4H AND,4H MIN
X
X
X

```



```

X      ,4HUS      ,4HSIGN,4H AS ,4HAPPR,4HOPRI,4HATE.,4H TH,4HE BO
X      ,4HX      ,4HMUST,4H CON,4HTAIN,4H SIG,4HN, D,4HECIM,4HAL A
X      ,4HND      ,4HNUMB,4HER.,4H NOT,4HE: T,4HHE M,4HAGNI,4HTUDE
X      ,4H OF      ,4HTHE ,4HZERO,4H MUS,4HT BE,4H LES,4HS TH,4HEN T /
DATA LTRARZ(64)/4HEN. /
CALL CHCKSZ(1,1)
IF( IERROR.NE.0) RETURN
CALL ERASE
CALL MOVABS(0,528)
DO 30 I=1,57,8
CALL ANMODE
WRITE(6,100)(LTRARZ(I+J-1),J=1,8)
CALL RECOVR
CALL NEWLIN
FORMAT(21X,8A4)
CALL NEWLIN
CALL ANMODE
WRITE(6,101)
FORMAT(27X,4HREAL)
CALL RECOVR
CALL NEWLIN
CALL MOVABS(320,302)
CALL DRWREL(0,22)
CALL DRWREL(138,0)
CALL DRWREL(0,-22)
CALL DRWREL(-138,0)
CALL MOVREL(0,3)
CALL CARTN
CALL ANMODE
WRITE(6,103)
FORMAT(22X,'>')
READ(5,104) X
FORMAT(F10.7)
CALL RECOVR
IF(ABS(X).GE.10.) IERROR=12
IF( IERROR.NE.0) RETURN
CALL NEWLIN
CALL NEWLIN
CALL ANMODE
Z=0.
WRITE(6,105) X,Z
FORMAT(15X,26HTHE NEW REAL ZERO WILL BE ,F10.7,3H +J,F10.7)
CALL RECOVR
CALL NEWLIN
CALL NEWLIN
WRITE(6,106)
FORMAT(21X,32HIF CORRECT TYPE 'Y', IF NOT 'N'.)
CALL RECOVR

```

ASDF3673  
ASDF3674  
ASDF3675  
ASDF3676  
ASDF3677  
ASDF3678  
ASDF3679  
ASDF3680  
ASDF3681  
ASDF3682  
ASDF3683  
ASDF3684  
ASDF3685  
ASDF3686  
ASDF3687  
ASDF3688  
ASDF3689  
ASDF3690  
ASDF3691  
ASDF3692  
ASDF3693  
ASDF3694  
ASDF3695  
ASDF3696  
ASDF3697  
ASDF3698  
ASDF3699  
ASDF3700  
ASDF3701  
ASDF3702  
ASDF3703  
ASDF3704  
ASDF3705  
ASDF3706  
ASDF3707  
ASDF3708  
ASDF3709  
ASDF3710  
ASDF3711  
ASDF3712  
ASDF3713  
ASDF3714  
ASDF3715  
ASDF3716  
ASDF3717  
ASDF3718  
ASDF3719  
ASDF3720



```

CALL SCURSR(ICHAR,IX,IY)
IF(ICHAR.NE.121.AND.ICHAR.NE.89)GO TO 10
CALL NUC
CALL STRSTS(1,1,X,0.)
RETURN
END

```

```

SUBROUTINE ARPRCT
COMMON IXXXXX(4),POLZRO(20,5),IERROR
DIMENSION LTRARP(64)
DATA LTRARP/4HTO A,4HDD A,4H REA,4HL PO,4HLE T,4HO TH,4HE DI
,4HS-,4HPLAY,4H ENT,4HER T,4HHE R,4HEAL,4HPARI,4H WIT
,4HHIN,4HTHE,4HBOX,4HPROV,4HIDE,4H,4H TH,4H NU,4HMBER
,4HTO I,4HNCLU,4HDE A,4H DEC,4H MAL,4H AND,4H MIN
,4HUS,4HSIGN,4H AS,4HAPPR,4HOPRI,4HATE,4H TH,4HE BO
,4HX,4HMUST,4H CON,4HTAIN,4H SIG,4HN, D,4HECIM,4HAL A
,4HND,4HNUMB,4HER.,4H NOT,4HE: T,4HHE M,4HAGNI,4HTUDE
,4H OF,4HTHE,4HPOLE,4H MUS,4HT BE,4H ONE,4H OR,4HLESS/
,4H LTRARP(64)/4H.
DATA LTRARP(64)/4H.

```

```

CALL CHCKSZ(1,3)
IF(IERROR.NE.0)RETURN
CALL ERASE
CALL MOVABS(0,528)
DO 30 I=1,57,8
CALL ANMODE
WRITE(6,100)
CALL RECOVER
CALL NEWLIN
FORMAT(21X,8A4)
CALL NEWLIN
CALL ANMODE
WRITE(6,101)
FORMAT(27X,4HREAL)
CALL RECOVER
CALL NEWLIN
CALL MOVABS(320,302)
CALL DRWREL(0,22)
CALL DRWREL(138,0)
CALL DRWREL(0,-22)
CALL DRWREL(-138,0)
CALL MOVREL(0,3)
CALL CARTN
CALL ANMODE
WRITE(6,103)
FORMAT(22X,'>')
READ(5,104)X
FORMAT(F10.7)
CALL RECOVER

```

10

30  
100

101

103  
104

ASDF3721  
ASDF3722  
ASDF3723  
ASDF3724  
ASDF3725  
ASDF3726  
ASDF3727  
ASDF3728  
ASDF3729  
ASDF3730  
ASDF3731  
ASDF3732  
ASDF3733  
ASDF3734  
ASDF3735  
ASDF3736  
ASDF3737  
ASDF3738  
ASDF3739  
ASDF3740  
ASDF3741  
ASDF3742  
ASDF3743  
ASDF3744  
ASDF3745  
ASDF3746  
ASDF3747  
ASDF3748  
ASDF3749  
ASDF3750  
ASDF3751  
ASDF3752  
ASDF3753  
ASDF3754  
ASDF3755  
ASDF3756  
ASDF3757  
ASDF3758  
ASDF3759  
ASDF3760  
ASDF3761  
ASDF3762  
ASDF3763  
ASDF3764  
ASDF3765  
ASDF3766  
ASDF3767  
ASDF3768





```

IF(ABS(X)-GT.1.0)IERROR=13
IF(IERROR.NE.0)CALL ERROR
IF(IERROR.NE.0)RETURN
CALL NEWLIN
CALL NEWLIN
CALL ANMODE
Z=0

```

```

105 WRITE(6,105) X,Z
FORMAT(15X,26HTHE NEW REAL POLE WILL BE ,F10.7,3H +J,F10.7)
CALL RECOVER
CALL NEWLIN
CALL ANMODE

```

```

106 WRITE(6,106)
FORMAT(21X,32HIF CORRECT TYPE 'Y', IF NOT 'N'.)
CALL RECOVER(ICHAR,IX,IY)
CALL SCURSR(ICHAR.NE.121.AND.ICHAR.NE.89)GO TO 10
IF(ICHAR.NE.121.AND.ICHAR.NE.89)GO TO 10
CALL NUC
CALL STRTS(3,1,X,0.)
RETURN
END

```

```

SUBROUTINE DCZRCI
COMMON IXXX(4),POLZRO(20,5),IERROR
DIMENSION LTRDCZ(64)
DATA LTRDCZ/4H D,4HELET,4HE A,4HCOMP,4HLEX,4HZERO,4H FRO
,4HM,4HTHE,4HDISP,4HLAY,,4H ENT,4HER T,4HHE,4HX, A
,4HND,4HY,4HCOOR,4HDINA,4HTES,4HOF A,4H POI,4HNT N
,4HEAR,4HONE,4HOF T,4HHE R,4HOOIS,4H TO,4HBE D,4HELET
,4HED,4HENTE,4HRT,4HHESE,4HVALU,4HES W,4HITHI,4HNT TH
,4HE,4HBOXE,4HS, I,4HNCUL,4HDE A,4H DEC,4HIMAL,4H, AN
,4HMNU,4HS SI,4HGN A,4HS AP,4HPROP,4HRIAT,4HE.
,4HNO,4HSIGN,4H IND,4HICAT,4HES P,4HOSIT,4HIVE,4HVALU/
DATA LTRDCZ(64)/4HES. /
CALL CHCKSZ(-1,2)
IF(IERROR.NE.0)RETURN
CALL ERASE(0,528)
CALL MOVABS(0,528)
DO 30 I=1,57,8
CALL ANMODE
WRITE(6,100)(LTRDCZ(I+J-1),J=1,8)
CALL NEWLIN
FORMAT(21X,8A4)
CALL NEWLIN
CALL ANMODE
WRITE(6,101)
FORMAT(29X,'X',12X,'Y')

```

```

10
30
100
101

```

ASDF3769  
ASDF3770  
ASDF3771  
ASDF3772  
ASDF3773  
ASDF3774  
ASDF3775  
ASDF3776  
ASDF3777  
ASDF3778  
ASDF3779  
ASDF3780  
ASDF3781  
ASDF3782  
ASDF3783  
ASDF3784  
ASDF3785  
ASDF3786  
ASDF3787  
ASDF3788  
ASDF3789  
ASDF3790  
ASDF3791  
ASDF3792  
ASDF3793  
ASDF3794  
ASDF3795  
ASDF3796  
ASDF3797  
ASDF3798  
ASDF3799  
ASDF3800  
ASDF3801  
ASDF3802  
ASDF3803  
ASDF3804  
ASDF3805  
ASDF3806  
ASDF3807  
ASDF3808  
ASDF3809  
ASDF3810  
ASDF3811  
ASDF3812  
ASDF3813  
ASDF3814  
ASDF3815  
ASDF3816





ASDF3817  
ASDF3818  
ASDF3819  
ASDF3820  
ASDF3821  
ASDF3822  
ASDF3823  
ASDF3824  
ASDF3825  
ASDF3826  
ASDF3827  
ASDF3828  
ASDF3829  
ASDF3830  
ASDF3831  
ASDF3832  
ASDF3833  
ASDF3834  
ASDF3835  
ASDF3836  
ASDF3837  
ASDF3838  
ASDF3839  
ASDF3840  
ASDF3841  
ASDF3842  
ASDF3843  
ASDF3844  
ASDF3845  
ASDF3846  
ASDF3847  
ASDF3848  
ASDF3849  
ASDF3850  
ASDF3851  
ASDF3852  
ASDF3853  
ASDF3854  
ASDF3855  
ASDF3856  
ASDF3857  
ASDF3858  
ASDF3859  
ASDF3860  
ASDF3861  
ASDF3862  
ASDF3863  
ASDF3864

```

103 CALL RECOVER
    CALL NEWLIN (320, 302)
    CALL MOVABS (0, 22)
    CALL DRWREL (138, 0)
    CALL DRWREL (0, -22)
    CALL DRWREL (-138, 0)
    CALL MOVREL (184, 0)
    CALL DRWREL (0, 22)
    CALL DRWREL (140, 0)
    CALL DRWREL (0, -22)
    CALL DRWREL (-140, 0)
    CALL MOVREL (0, 3)
    CALL CARTN
    CALL ANMODE
    WRITE (6, 103)
    FORMAT (22X, '>')
    READ (5, 104) X, Y
    FORMAT (F10.7, 5X, F10.7)
    CALL RECOVER
    CALL LOCATE (2, X, Y, LINEPZ)
    CALL NEWLIN
    CALL NEWLIN
    CALL ANMODE
    WRITE (6, 105) POLZRO(LINEPZ, 1), POLZRO(LINEPZ, 2)
    FORMAT (I1X, 26HDELETED ZERO PAIR WILL BE, F10.7, 6H +/- J, F10.7)
    CALL RECOVER
    CALL NEWLIN
    CALL ANMODE
    WRITE (6, 106)
    FORMAT (21X, 32HIF CORRECT TYPE 'Y', IF NOT 'N'.)
    CALL RECOVER
    CALL SCURSR (ICAR, IX, IV)
    IF (ICAR.NE.121.AND.ICAR.NE.89) GO TO 10
    CALL UPDATE (2, LINEPZ)
    CALL NUC
    RETURN
END

SUBROUTINE DCPRCT
COMMON IXX(4), POLZRO(20, 5), IERROR
DIMENSION LTRDCP(64)
DATA LTRDCP/4HTO D, 4HELET, 4HE A, 4HCOMP, 4HLEX, 4HPOLE, 4H FRO
, 4HM, 4HTHE, 4HUNIT, 4HCIR, 4HCLE, 4H ENT, 4HER I, 4HHE P
, 4HX, 4HAND, 4HY, 4HCOOR, 4HDINA, 4HTES, 4HOF T, 4HHE P
, 4HOINT, 4HNEAR, 4H ONE, 4H OF, 4HTHE, 4HROOT, 4HS TO, 4H BE
, 4HDEL-, 4HETED, 4H. EN, 4HTER, 4HTHE, 4HVALU, 4HES I, 4HN TH
, 4HE, 4HBOXE, 4HS. I, 4HNCUL, 4HDE A, 4H DEC, 4HIMAL, 4H, AN
X X X X X

```



```

X      ,4HND      ,4HMINU,4HS  SI,4HGN A,4HS  AP,4HPROP,4HRIAT,4HE.
X      ,4HNO      ,4HSIGN,4H  IND,4HICAT,4HES P,4HOSIT,4HIVE ,4HVALU/
DATA  LTRDCP(64)/4HES. /
CALL  CHCKSZ(-1,4)
IF(IERROR.NE.0)RETURN
CALL  ERASE
CALL  MOVABS(0,528)
DO 30 I=1,57,8
CALL  ANMODE
WRITE(6,100)(LTRDCP(I+J-1),J=1,8)
CALL  RECOVR
CALL  NEWLIN
CALL  NEWLIN
CALL  ANMODE
WRITE(6,101)
FORMAT(21X,'8A4')
CALL  RECOVR
CALL  NEWLIN
CALL  MOVABS(320,302)
CALL  DRWREL(0,22)
CALL  DRWREL(138,0)
CALL  DRWREL(0,-22)
CALL  DRWREL(-138,0)
CALL  MOVREL(184,0)
CALL  DRWREL(0,22)
CALL  DRWREL(140,0)
CALL  DRWREL(0,-22)
CALL  DRWREL(-140,0)
CALL  MOVREL(0,3)
CALL  CARTN
CALL  ANMODE
WRITE(6,103)
FORMAT(22X,'>')
READ(5,104) X,Y
FORMAT(F10.7,5X,F10.7)
CALL  RECOVR
CALL  LOCATE(4,X,Y,LINEPZ)
CALL  NEWLIN
CALL  NEWLIN
CALL  ANMODE
WRITE(6,105)POLZRO(LINEPZ,1),POLZRO(LINEPZ,2)
FORMAT(11X,26HDELETED POLE PAIR WILL BE ,F10.7,6H +/- J,F10.7)
CALL  RECOVR
CALL  LOCATE(4,X,Y,LINEPZ)
CALL  NEWLIN
CALL  ANMODE
WRITE(6,106)

```



106

```

FORMAT(21X,32HIF CORRECT TYPE 'Y', IF NOT 'N'.)
CALL RECOVER
CALL SCURSR(ICHAR,IX,IY)
IF(ICHAR.NE.121.AND.ICHAR.NE.89)GO TO 10
CALL UPDATE(4,LINEPZ)
CALL NUC
RETURN
END

```

```

SUBROUTINE DRZRCI
COMMON IXXXX(4),POLZRO(20,5),IERROR
DIMENSION LTRDRZ(64)
DATA LTRDRZ/4H10 D,4HELET,4HE A,4HREAL,4H ZER,4HO FR,4HOM T
,4HHE ,4HDISP,4HLAY,4H ENT,4HER T,4HHE ,4HX C,4HOORD
,4HIN- ,4HATE ,4HOF A,4H POI,4HNT N,4HEAR T,4HHE V,4HVALUE
,4H TO ,4HBE D,4HELET,4HED. B,4HOX P,4HROVI,4HDEDE.,4H INC
,4H ,4HWITH,4HIN T,4HHE ,4HND M,4HINUS,4H SIG,4HN AS
,4HLUDE ,4HA DE,4HCIMA,4HL, A,4HND NO,4H SIG,4HN IN,4H DICA
,4H ,4HTES ,4HAPPR,4HOPRI,4HATE.,4H ,4H
,4H LTRDRZ(64)/4H
DATA LTRDRZ(64)/4H
CALL CHCKSZ(-1,1)
IF(IERROR.NE.0)RETURN
CALL ERASE(0,528)
DO 30 I=1,57,8
CALL ANMODE
WRITE(6,100)(LTRDRZ(I+J-1),J=1,8)
CALL RECOVER
CALL NEWLIN
FORMAT(21X,8A4)
CALL ANMODE
WRITE(6,101)
FORMAT(28X,'X')
CALL RECOVER
CALL NEWLIN
CALL MOVABS(320,302)
CALL DRWREL(0,22)
CALL DRWREL(138,0)
CALL DRWREL(0,-22)
CALL DRWREL(-138,0)
CALL MOVREL(0,3)
CALL CARTN
CALL ANMODE
WRITE(6,103)
FORMAT(22X,'>')

```

10

30  
100

101

103

ASDF3913  
ASDF3914  
ASDF3915  
ASDF3916  
ASDF3917  
ASDF3918  
ASDF3919  
ASDF3920  
ASDF3921  
ASDF3922  
ASDF3923  
ASDF3924  
ASDF3925  
ASDF3926  
ASDF3927  
ASDF3928  
ASDF3929  
ASDF3930  
ASDF3931  
ASDF3932  
ASDF3933  
ASDF3934  
ASDF3935  
ASDF3936  
ASDF3937  
ASDF3938  
ASDF3939  
ASDF3940  
ASDF3941  
ASDF3942  
ASDF3943  
ASDF3944  
ASDF3945  
ASDF3946  
ASDF3947  
ASDF3948  
ASDF3949  
ASDF3950  
ASDF3951  
ASDF3952  
ASDF3953  
ASDF3954  
ASDF3955  
ASDF3956  
ASDF3957  
ASDF3958  
ASDF3959  
ASDF3960





ASDF3961  
ASDF3962  
ASDF3963  
ASDF3964  
ASDF3965  
ASDF3966  
ASDF3967  
ASDF3968  
ASDF3969  
ASDF3970  
ASDF3971  
ASDF3972  
ASDF3973  
ASDF3974  
ASDF3975  
ASDF3976  
ASDF3977  
ASDF3978  
ASDF3979  
ASDF3980  
ASDF3981  
ASDF3982  
ASDF3983  
ASDF3984  
ASDF3985  
ASDF3986  
ASDF3987  
ASDF3988  
ASDF3989  
ASDF3990  
ASDF3991  
ASDF3992  
ASDF3993  
ASDF3994  
ASDF3995  
ASDF3996  
ASDF3997  
ASDF3998  
ASDF3999  
ASDF4000  
ASDF4001  
ASDF4002  
ASDF4003  
ASDF4004  
ASDF4005  
ASDF4006  
ASDF4007  
ASDF4008

```

104 READ (5,104) X
    FORMAT(F10.7)
    CALL RECOVER
    CALL LOCATE(1,X,0.,LINEPZ)
    CALL NEWLIN
    CALL ANMODE
    Z=0.
105 WRITE(6,105)POLZRO(LINEPZ,1),POLZRO(LINEPZ,2)
    FORMAT(15X,26HDELETED REAL ZERO WILL BE ,F10.7,3H +J,F10.7)
    CALL RECOVER
    CALL NEWLIN
    CALL ANMODE
106 WRITE(6,106)
    FORMAT(21X,32HIF CORRECT TYPE 'Y', IF NOT 'N'.)
    CALL RECOVER
    CALL SCURSR(ICHAR,IX,IY)
    IF(ICHAR.NE.121.AND.ICHAR.NE.89)GO TO 10
    CALL UPDATE(1,LINEPZ)
    CALL NUC
    RETURN
END

SUBROUTINE DRPRCT POLZRO(20,5), IERROR
COMMON IXXXXX(4), POLZRO(64)
DIMENSION LTRDRP(64)
DATA LTRDRP/4HTO D,4HELET,4HE A ,4HREAL,4H POL,4HE FR,4HOM T
,4HHE ,4HUNIT,4H CIR,4HCLE,,4H ENT,4HER T,4HHE ,4HX, C
,4HOOR-,4HDINA,4HTE O,4HFA ,4HPOIN,4HT NE,4HAR T,4HHE R
,4HOOT ,4HTO B,4HE DE,4HLETE,4HDE. E,4HNTER,4H THE,4H VAL
,4HUE ,4HWI TH,4HIN T,4HHE B,4HDX P,4HROVI,4HDEED.,4H INC
,4HLUDE,4HA DE,4HCIMA,4HL, A,4HND M,4HINUS,4H SIG,4HNAS
,4H ,4H ,4HAPPR,4HOPRI,4HATE.,4H NO,4H SIG,4HIN,4HUICA ,
,4HTES ,4HA PO,4HSITI,4HVE V,4HALUE,4H. ,4H
DATA LTRDRP(64)/4H
CALL CHCKSZ(-1,3)
IF(IERROR.NE.0)RETURN
CALL ERASE
CALL MOVABS(0,528)
DO 30 I=1,57,8
CALL ANMODE
WRITE(6,100)(LTRDRP(I+J-1),J=1,8)
CALL RECOVER
CALL NEWLIN
FORMAT (21X,8A4)
CALL NEWLIN
CALL ANMODE
WRITE (6,101)

```





ASDF4009  
ASDF4010  
ASDF4011  
ASDF4012  
ASDF4013  
ASDF4014  
ASDF4015  
ASDF4016  
ASDF4017  
ASDF4018  
ASDF4019  
ASDF4020  
ASDF4021  
ASDF4022  
ASDF4023  
ASDF4024  
ASDF4025  
ASDF4026  
ASDF4027  
ASDF4028  
ASDF4029  
ASDF4030  
ASDF4031  
ASDF4032  
ASDF4033  
ASDF4034  
ASDF4035  
ASDF4036  
ASDF4037  
ASDF4038  
ASDF4039  
ASDF4040  
ASDF4041  
ASDF4042  
ASDF4043  
ASDF4044  
ASDF4045  
ASDF4046  
ASDF4047  
ASDF4048  
ASDF4049  
ASDF4050  
ASDF4051  
ASDF4052  
ASDF4053  
ASDF4054  
ASDF4055  
ASDF4056

```

101  FORMAT(28X,'X')
      CALL RECOVER
      CALL NEWLIN
      CALL MOVABS(320,302)
      CALL DRWREL(0,22)
      CALL DRWREL(138,0)
      CALL DRWREL(0,-22)
      CALL DRWREL(-138,0)
      CALL MOVREL(0,3)
      CALL CARTN
      CALL ANMODE
      WRITE(6,103)
103  FORMAT(22X,'>')
      READ(5,104) X
104  FORMAT(10,7)
      CALL RECOVER
      CALL LOCATE(3,X,0.,LINEPZ)
      CALL NEWLIN
      CALL ANMODE
      WRITE(6,105)
105  POLZRO(LINEPZ,1),POLZRO(LINEPZ,2)
      FORMAT(15X,26HDELETED REAL POLE WILL BE ,F10.7,3H +J,F10.7)
      CALL RECOVER
      CALL NEWLIN
      CALL ANMODE
      WRITE(6,106)
106  FORMAT(21X,32HIF CORRECT TYPE 'Y', IF NOT 'N'.)
      CALL RECOVER(ICHAR,IX,IY)
      IF(ICHAR.NE.121.AND.(CHAR.NE.89)GO TO 10
      CALL UPDATE(3,LINEPZ)
      CALL NUC
      RETURN
      END

SUBROUTINE ACZPLR
COMMON IXXXX(4),POLZRO(20,5),IERROR
DIMENSION LTPACZ(64)
DATA LTPACZ/4H D A,4H DD A,4H COM,4H PLEX,4H ZER,4H O TO,4H THE
X      ,4H ,4H DIS,4H LAY,,4H ENT,4H ER T,4H HE R,4H ADIA,4H L AN
X      ,4H HD,4H THET,4H A CO,4H MPON,4H ENIS,4H WIT,4H HIN ,4H PROV
X      ,4H HDED,4H BOXE,4H S T,4H HE M,4H AGNI,4H TUDE,4H OF ,4H RHO
X      ,4H MUST,4H BE L,4H ESS ,4H THAN,4H TEN,4H TH,4H ETA(,4H RADI
X      ,4H ANS),4H EACH,4H NUM,4H BER ,4H REQU,4H I RES,4H A D,4H ECI M
X      ,4H AL,4H AND ,4H AS A,4H PPRO,4H PRIA,4H TE A,4H MIN,4H US S
X      ,4H IGN,4H BOTH,4H ARE,4H CON,4H FINE,4H D TO,4H THE,4H BOX/
      DATA LTPACZ(64)/4HES. /
      CALL CHCKSZ(2,2)

```



|     |   |          |
|-----|---|----------|
| 10  | IF(IERROR.NE.0)RETURN   | ASDF4057 |
|     | CALL ERASE  | ASDF4058 |
|     | CALL MOVABS(0,528)  | ASDF4059 |
|     | DO 30 I=1,57,8  | ASDF4060 |
|     | CALL ANMODE   | ASDF4061 |
|     | WRITE(6,100)(LTPACZ(I+J-1),J=1,8)                                 | ASDF4062 |
|     | CALL RECOVER  | ASDF4063 |
|     | CALL NEWLIN   | ASDF4064 |
| 30  | FORMAT(21X,8A4)   | ASDF4065 |
| 100 | CALL NEWLIN   | ASDF4066 |
|     | CALL ANMODE   | ASDF4067 |
|     | WRITE(6,101)  | ASDF4068 |
| 101 | FORMAT(25X,'MAGNITUDE',8X,'ANGLE')                                | ASDF4069 |
|     | CALL RECOVER  | ASDF4070 |
|     | CALL NEWLIN   | ASDF4071 |
|     | CALL ANMODE   | ASDF4072 |
|     | WRITE(6,102)  | ASDF4073 |
| 102 | FORMAT(34X,3H+/-)   | ASDF4074 |
|     | CALL RECOVER  | ASDF4075 |
|     | CALL ANGLE(504,304)   | ASDF4076 |
|     | CALL MOVABS(320,302)  | ASDF4077 |
|     | CALL DRWREL(0,22)   | ASDF4078 |
|     | CALL DRWREL(138,0)  | ASDF4079 |
|     | CALL DRWREL(0,-22)  | ASDF4080 |
|     | CALL DRWREL(-138,0)   | ASDF4081 |
|     | CALL MDVREL(212,0)  | ASDF4082 |
|     | CALL DRWREL(0,22)   | ASDF4083 |
|     | CALL DRWREL(140,0)  | ASDF4084 |
|     | CALL DRWREL(0,-22)  | ASDF4085 |
|     | CALL DRWREL(-140,0)   | ASDF4086 |
|     | CALL MDVREL(0,3)  | ASDF4087 |
|     | CALL CARTN  | ASDF4088 |
|     | CALL ANMODE   | ASDF4089 |
|     | WRITE(6,103)  | ASDF4090 |
| 103 | FORMAT(22X,'>')   | ASDF4091 |
|     | READ(5,104) RHO,THETA   | ASDF4092 |
| 104 | FORMAT(F10.7,5X,F10.7)  | ASDF4093 |
|     | CALL RECOVER  | ASDF4094 |
|     | IF(RHO.GE.1)IERROR=14   | ASDF4095 |
|     | IF(IERROR.NE.0)CALL ERROR   | ASDF4096 |
|     | IF(IERROR.NE.0)RETURN   | ASDF4097 |
|     | CALL NEWLIN   | ASDF4098 |
|     | CALL ANMODE   | ASDF4099 |
|     | THETA=ABS(THETA)  | ASDF4100 |
|     | WRITE(6,105) RHO,THETA  | ASDF4101 |
| 105 | FORMAT(9X,30H THE NEW COMPLEX ZEROS WILL BE ,F10.7,6H +/- ,F10.7) | ASDF4102 |
|     | CALL RECOVER  | ASDF4103 |
|     |   | ASDF4104 |



ASDF4105  
ASDF4106  
ASDF4107  
ASDF4108  
ASDF4109  
ASDF4110  
ASDF4111  
ASDF4112  
ASDF4113  
ASDF4114  
ASDF4115  
ASDF4116  
ASDF4117  
ASDF4118  
ASDF4119  
ASDF4120  
ASDF4121  
ASDF4122  
ASDF4123  
ASDF4124  
ASDF4125  
ASDF4126  
ASDF4127  
ASDF4128  
ASDF4129  
ASDF4130  
ASDF4131  
ASDF4132  
ASDF4133  
ASDF4134  
ASDF4135  
ASDF4136  
ASDF4137  
ASDF4138  
ASDF4139  
ASDF4140  
ASDF4141  
ASDF4142  
ASDF4143  
ASDF4144  
ASDF4145  
ASDF4146  
ASDF4147  
ASDF4148  
ASDF4149  
ASDF4150  
ASDF4151  
ASDF4152

```

CALL ANGLE(727,260)
CALL NEWLIN
CALL ANMODE
WRITE(6,106)
FORMAT(21X,32HIF CORRECT TYPE 'Y', IF NOT 'N'.)
CALL RECOVER
CALL SCURSR(ICHAR,IX,IY)
IF(ICHAR.NE.121.AND.ICHAR.NE.89)GO TO 10
CALL NUC
CALL STRRTS(2,2,RHO,THETA)
RETURN
END

SUBROUTINE ACPLR
COMMON IXXX(4),POLZRO(20,5),IERROR
DIMENSION LTPACP(64)
DATA LTPACP/4HTO A,4HDD A,4H COM,4HPLEX,4H POL,4HE WI,4HTHIN
,4H THE,4HUNIT,4H CIR,4HCLE,4HENTIE,4HHR TH,4HE RA,4HDIAL
,4H AND,4HTHET,4H CO,4HMPON,4HENTS,4H WIT,4HHIN,4HPROV
,4HDED,4HBOXE,4HS,4HRO,4HMUST,4H BE,4HONE,4HOR L
,4HESS,4HTHET,4H MU,4HST B,4HE EN,4HTERE,4H ID,4HECIM
,4HIANS,4HEACH,4H NUM,4HBER,4HREQU,4HIRE A,4H MIN,4HUS S
,4HAL,4HAND,4HAS A,4HPPRO,4HPRIA,4HTE TO,4H THE,4H BOX/
,4HIGN,4HBOOTH,4H ARE,4H CON,4HFINE,4H TO,4H THE,4H BOX/
DATA LTPACP(64)/4HES. /
CALL CHCKSZ(2,4)
IF(IERROR.NE.0)RETURN
CALL ERASE
CALL MOVABS(0,528)
DO 30 I=1,57,8
CALL ANMODE
WRITE(6,100)(LTPACP(I+J-1),J=1,8)
CALL RECOVER
CALL NEWLIN
FORMAT(21X,8A4)
CALL NEWLIN
CALL ANMODE
WRITE(6,101)
FORMAT(25X,'MAGNITUDE',8X,'ANGLE')
CALL RECOVER
CALL NEWLIN
CALL ANMODE
WRITE(6,102)
FORMAT(34X,3H+/-)
CALL RECOVER
CALL ANGLE(504,304)
CALL MOVABS(320,302)
CALL DWREL(0,22)

```





```

CALL DRWREL(138,0)
CALL DRWREL(0,-22)
CALL DRWREL(-138,0)
CALL MOVREL(212,0)
CALL DRWREL(0,22)
CALL DRWREL(140,0)
CALL DRWREL(0,-22)
CALL DRWREL(-140,0)
CALL CARIN
CALL ANMODE
WRITE(6,103)
FORMAT(22X,'>')
READ(5,104) RHO,THETA
FORMAT(F10.7,5X,F10.7)
CALL RECOVER
IF(RHO.GT.1.0) IERROR=15
IF(IERROR.NE.0) RETURN
CALL NEWLIN
CALL ANMODE
THETA=ABS(THETA)
WRITE(6,105) RHO,THETA
FORMAT(9X,30H THE NEW COMPLEX POLES WILL BE ,F10.7,6H +/- ,F10.7)
CALL RECOVER(727,260)
CALL NEWLIN
CALL ANMODE
WRITE(6,106)
FORMAT(21X,32H IF CORRECT TYPE 'Y', IF NOT 'N'.)
CALL RECOVER
CALL SCURSR(ICHAR,IX,IY)
IF(ICHAR.NE.121.AND.ICHAR.NE.89) GO TO 10
CALL NUC
CALL STRSTS(4,2,RHO,THETA)
RETURN
END

SUBROUTINE ARZPLR
COMMON IXXX(4),POLZRO(20,5),IERROR
DIMENSION LTPARZ(64)
DATA LTPARZ/4HTD A,4HDD A,4H REA,4HL ZE,4HRO T,4HO TH,4HE DI
      4HS-
      4H IN,4HTHE,4H EN,4HTIER,4HTHE,4HVALU,4HE OF,4H RHO
      4HUDE,4HOF R,4HHD M,4HUST,4HBE L,4HESS,4HTHAN,4H TEN
      4HD,4H THE,4HNUMB,4HER,4HINCL,4HUDIN,4HGD,4HCIMA,4HL AN
      4HCON-,4HTAIN,4HED W,4HTITHI,4HN TH,4HE BO,4HX.,4HNO S
X
X
X
X
X
X
X

```





```

X      ,4HIGN ,4HINDI,4HCATE,4HS A ,4HPOSI,4HTIVE,4H VAL,4HUE. /
DATA LTPARZ(64)/4H
CALL CHCKSZ(1,1)
IF(IEERROR.NE.0)RETURN
CALL ERASE(0,528)
DO 30 I=1,578
CALL ANMODE
WRITE(6,100)(LTPARZ(I+J-1),J=1,8)
CALL RECOVR
CALL NEWLIN
FORMAT(21X,8A4)
CALL ANMODE
WRITE(6,101)
FORMAT(25X,'MAGNITUDE')
CALL RECOVR
CALL NEWLIN
CALL MOVABS(320,302)
CALL DRWREL(0,22)
CALL DRWREL(138,0)
CALL DRWREL(0,-22)
CALL DRWREL(-138,0)
CALL MOVREL(0,3)
CALL CARTN
CALL ANMODE
WRITE(6,103)
FORMAT(22X,'>')
READ(5,104) RHO
FORMAT(F10.7)
CALL RECOVR
IF(RHO.GT.1.0)IEERROR=16
IF(IEERROR.NE.0)RETURN
CALL NEWLIN
CALL ANMODE
THETA=0.
WRITE(6,105) RHO,THETA
FORMAT(12X,26H)THE NEW REAL ZERO WILL BE ,F10.7,3X,F10.7)
CALL RECOVR(677,260)
CALL NEWLIN
CALL ANMODE
WRITE(6,106)
FORMAT(21X,32H)IF CORRECT TYPE 'Y', IF NOT 'N'.)
CALL RECOVR(ICHAR,IX,IY)
CALL SCURSR(ICHAR.NE.121.AND.ICHAR.NE.89)GO TO 10
IF(ICHAR.NE.121.AND.ICHAR.NE.89)GO TO 10

```

10

30  
100

101

103

104

105

106



ASDF4249  
ASDF4250  
ASDF4251  
ASDF4252  
ASDF4253  
ASDF4254  
ASDF4255  
ASDF4256  
ASDF4257  
ASDF4258  
ASDF4259  
ASDF4260  
ASDF4261  
ASDF4262  
ASDF4263  
ASDF4264  
ASDF4265  
ASDF4266  
ASDF4267  
ASDF4268  
ASDF4269  
ASDF4270  
ASDF4271  
ASDF4272  
ASDF4273  
ASDF4274  
ASDF4275  
ASDF4276  
ASDF4277  
ASDF4278  
ASDF4279  
ASDF4280  
ASDF4281  
ASDF4282  
ASDF4283  
ASDF4284  
ASDF4285  
ASDF4286  
ASDF4287  
ASDF4288  
ASDF4289  
ASDF4290  
ASDF4291  
ASDF4292  
ASDF4293  
ASDF4294  
ASDF4295  
ASDF4296

```

CALL NUC
CALL STRTS(1,2,RHO,0.)
RETURN
END

SUBROUTINE ARPPLR
COMMON IXXX(4),POLZRO(20,5),IERROR
DIMENSION LTPARP(64)
DATA LTPARP/4HTO A,4HDD A,4H REA,4HL PO,4HLE T,4HO TH,4HE DI
,4HS- ,4HPLAY,4H ENT,4HER A,4H VAL,4HUE O,4HF RH,4HO,
,4HONE ,4HOR L,4HMESS,4H IN ,4HTHE ,4HBOX ,4HPR OV,4HIDE
,4H. ,4HHE ,4HNUMB,4HER, ,4HINCL,4HUDIN,4HG DE,4HCIMA
,4HL ,4HAND ,4HSIGN,4H AS ,4HAPPR,4HOPRI,4HATE ,4HMUST
,4H BE ,4HCONT,4HAIN,4HD WI,4HTHIN,4H THE,4H BOX,4H.
,4H ,4HNOTE,4H: TH,4HE MA,4HGNIT,4HUDE ,4HOF R,4HHO M
,4HUST ,4HBE O,4HNE O,4HR LE,4HSS. ,4H ,4H
,4H LTPARP(64)/4H
DATA CHCKSZ(1,3)
CALL CHCKSZ(1,3)
IF(IERROR.NE.0)RETURN
CALL ERASE
CALL MOVABS(0,528)
DO 30 I=1,57,8
CALL ANMODE
WRITE(6,100)(LTPARP(I+J-1),J=1,8)
CALL RECOVR
CALL NEWLIN
FORMAT(21X,8A4)
CALL ANMODE
CALL ANMODE(6,101)
WRITE(6,101)MAGNITUDE*)
FORMAT(25X,MAGNITUDE*)
CALL RECOVR
CALL NEWLIN
CALL MOVABS(320,302)
CALL DRWREL(0,22)
CALL DRWREL(138,0)
CALL DRWREL(0,-22)
CALL DRWREL(-138,0)
CALL MOVREL(0,3)
CALL CARIN
CALL ANMODE
WRITE(6,103)
FORMAT(22X,'>')
READ(5,104)RHO
FORMAT(F10.7)
CALL RECOVR
IF(RHO.GT.1.0)IERROR=17
IF(IERROR.NE.0)RETURN

```

10

30  
100

101

103

104



```

CALL NEWLIN
CALL NEWLIN
CALL ANMODE
THETA=0.
WRITE(6,105) RHO, THETA
FORMAT(12X, 26) THE NEW REAL POLE WILL BE , F10.7, 3X, F10.7)
CALL RECOVER
CALL ANGLE(677, 260)
CALL NEWLIN
CALL ANMODE
WRITE(6,106)
FORMAT(21X, 32) IF CORRECT TYPE 'Y', IF NOT 'N'.)
CALL RECOVER
CALL SCURSR(ICHAR, IX, IY)
IF(ICHAR.NE.121.AND. ICHAR.NE.89) GO TO 10
CALL NUC
CALL STRRTS(3, 2, RHO, 0.)
RETURN
END

SUBROUTINE DCZPLR
COMMON IXX(4), POLZRO(20, 5), IERROR
DIMENSION LTPDCZ(64)
DATA LTPDCZ/4H 0 D, 4HELET, 4HE A, 4HCOMP, 4HLEX, 4HZERO, 4H FRO
X 4HM AND, 4HTHE, 4HDISP, 4HCLAY, 4H ENT, 4HER T, 4HHE, 4HHR HO,
X 4HNT, 4H THE, 4HTA, 4HCOOR, 4HDINA, 4HTES, 4HOF A, 4H POI
X 4HDEL, 4HNEAR, 4H ONE, 4H OF, 4HTHE, 4HROOT, 4HSTO, 4H BE
X 4HTHE, 4HED, 4HEN, 4HTER, 4HTHE, 4HVA, 4HLCUES, 4H IN
X 4HND, 4HBOXE, 4HS. EN, 4HINCL, 4HUDE, 4HA DE, 4HCIMA, 4HL, A
X 4HNO, 4HMUNU, 4HS SI, 4HGN A, 4HS AP, 4HPROP, 4HRIAT, 4HE.
X 4HNO, 4HSIGN, 4H IND, 4HICAT, 4HES P, 4HOSIT, 4HIVE, 4HVALU/
DATA LTPDCZ(64)/4HES. /
CALL CHCKSZ(-1, 2)
IF(IERROR.NE.0) RETURN
CALL ERASE
CALL MOVABS(0, 528)
DO 30 I=1, 57, 8
CALL ANMODE
WRITE(6, 100) (LTPDCZ(I+J-1), J=1, 8)
CALL RECOVER
CALL NEWLIN
FORMAT(21X, 8A4)
CALL NEWLIN
CALL ANMODE
WRITE(6, 101)
FORMAT(27X, 12X, 'THETA.')
CALL RECOVER
CALL NEWLIN

```

ASDF4297  
ASDF4298  
ASDF4299  
ASDF4300  
ASDF4301  
ASDF4302  
ASDF4303  
ASDF4304  
ASDF4305  
ASDF4306  
ASDF4307  
ASDF4308  
ASDF4309  
ASDF4310  
ASDF4311  
ASDF4312  
ASDF4313  
ASDF4314  
ASDF4315  
ASDF4316  
ASDF4317  
ASDF4318  
ASDF4319  
ASDF4320  
ASDF4321  
ASDF4322  
ASDF4323  
ASDF4324  
ASDF4325  
ASDF4326  
ASDF4327  
ASDF4328  
ASDF4329  
ASDF4330  
ASDF4331  
ASDF4332  
ASDF4333  
ASDF4334  
ASDF4335  
ASDF4336  
ASDF4337  
ASDF4338  
ASDF4339  
ASDF4340  
ASDF4341  
ASDF4342  
ASDF4343  
ASDF4344





```

102      CALL ANMODE
        WRITE (6,102)
        FORMAT (34X,3H+/-)
        CALL ANGLE(504,304)
        CALL RECOVR(320,302)
        CALL DRWREL(0,22)
        CALL DRWREL(138,0)
        CALL DRWREL(0,-22)
        CALL DRWREL(-138,0)
        CALL MOVREL(212,0)
        CALL DRWREL(0,22)
        CALL DRWREL(140,0)
        CALL DRWREL(0,-22)
        CALL DRWREL(-140,0)
        CALL MOVREL(0,3)
        CALL CARTN
        CALL ANMODE
        WRITE (6,103)
        FORMAT (22X,'>')
        READ (5,104) RHO, THETA
        FORMAT (F10.7,5X,F10.7)
103      CALL RECOVR
        X=RHO*COS(THETA)
        Y=RHO*SIN(THETA)
        CALL LOCATE(2,X,Y,LINEPZ)
        CALL NEWLIN
        CALL ANMODE
        WRITE(6,105)POLZRO(LINEPZ,3),POLZRO(LINEPZ,4)
        FORMAT(9X,30HDELETED COMPLEX ZEROS WILL BE ,F10.7,6H +/- ,F10.7)
104      CALL RECOVR
        CALL ANGLE(727,260)
        CALL NEWLIN
        CALL ANMODE
        WRITE(6,106)
        FORMAT(21X,32HIF CORRECT TYPE 'Y', IF NOT 'N'.)
105      CALL RECOVR
        CALL SCURSR(ICHAR,IX,IY)
        IF(ICHAR.NE.121.AND.ICHAR.NE.89)GO TO 10
        CALL UPDATE(2,LINEPZ)
        CALL NUC
        RETURN
        END

SUBROUTINE DCPPLR
COMMON IXXXX(4),POLZRO(20,5),IERROR
DIMENSION LTPDCP(64)

```





```

DATA LTPDCP/4HTO D,4HELET,4HE A,4HCOMP,4HLEX,4HPOLE,4H FRO
X,4HM,4HTHE,4HUNIT,4HCIR,4HCLE,4H ENT,4HER T,4HHE A
X,4HRHO,4HAND,4HTHE,4HTA,4HCOOR,4HDINA,4HTES,4HOF A
X,4HPOIN,4HTNE,4HAR O,4HNE O,4HTH,4HRO,4HOTS
X,4HTO,4HBE D,4HELET,4HED,4H ENT,4HER T,4HHE V,4HALUE
X,4HS IN,4HTHE,4HBOXE,4HS,4HINCL,4HUDE,4HA DE,4HCIMA
X,4HL,4HAND,4HMINU,4HS SI,4HGN A,4HS AP,4HPROP,4HRIAT
X,4HE,4HNO S,4HIGN,4HIMPL,4HIES,4HPOSI,4HTIVE,4H VAL/
DATA LTPDCP(64)/4HUES./
CALL CHCKSZ(-1,4)
IF(IERROR.NE.0)RETURN
CALL ERASE
CALL MOVABS(0,528)
DO 30 I=1,57,8
CALL ANMODE
WRITE(6,100)(LTPDCP(I+J-1),J=1,8)
CALL RECOVR
CALL NEWLIN
FORMAT(21X,8A4)
CALL ANMODE
WRITE(6,101)
FORMAT(27X,'RHO',12X,'THETA')
CALL RECOVR
CALL ANMODE
WRITE(6,102)
FORMAT(34X,3H+/-)
CALL RECOVR
CALL ANGLE(504,304)
CALL MOVABS(320,302)
CALL DRWREL(0,22)
CALL DRWREL(138,0)
CALL DRWREL(0,-22)
CALL DRWREL(-138,0)
CALL MOVREL(212,0)
CALL DRWREL(0,22)
CALL DRWREL(140,0)
CALL DRWREL(0,-22)
CALL DRWREL(-140,0)
CALL MOVREL(0,3)
CALL CARTRN
CALL ANMODE
WRITE(6,103)
FORMAT(22X,'>')
READ(5,104)RHO,THETA
FORMAT(F10.7,5X,F10.7)
CALL RECOVR

```

```

ASDF4393
ASDF4394
ASDF4395
ASDF4396
ASDF4397
ASDF4398
ASDF4399
ASDF4400
ASDF4401
ASDF4402
ASDF4403
ASDF4404
ASDF4405
ASDF4406
ASDF4407
ASDF4408
ASDF4409
ASDF4410
ASDF4411
ASDF4412
ASDF4413
ASDF4414
ASDF4415
ASDF4416
ASDF4417
ASDF4418
ASDF4419
ASDF4420
ASDF4421
ASDF4422
ASDF4423
ASDF4424
ASDF4425
ASDF4426
ASDF4427
ASDF4428
ASDF4429
ASDF4430
ASDF4431
ASDF4432
ASDF4433
ASDF4434
ASDF4435
ASDF4436
ASDF4437
ASDF4438
ASDF4439
ASDF4440

```

10

30  
100

101

102

103

104



```

X=RHO*COS(THETA)
Y=RHO*SIN(THETA)
CALL LOCATE(4,X,Y,LINEPZ)
CALL NEWLIN
CALL NEWLIN
CALL ANMODE
WRITE(6,105)POLZRO(LINEPZ,3),POLZRO(LINEPZ,4)
FORMAT(9X,29HDELETED COMPLEX POLE WILL BE ,F10.7,6H +/- ,F10.7)
105 CALL RECOVER
CALL ANGLE(727,260)
CALL NEWLIN
CALL ANMODE
WRITE(6,106)
106 FORMAT(21X,32HIF CORRECT TYPE 'Y', IF NOT 'N'.)
CALL RECOVER
CALL SCURSR(ICHAR,IX,IY)
IF(ICHAR.NE.121.AND.ICHAR.NE.89)GO TO 10
CALL UPDATE(4,LINEPZ)
CALL NUC
RETURN
END
SUBROUTINE DRZPLR
COMMON IXXX(4),POLZRO(20,5),IERROR
DIMENSION LTPDRZ(64)
DATA LTPDRZ/4HIO D,4HELET,4HE A,4HREAL,4H ZER,4HD FR,4HOM T
X,4HHE,4HDI SP,4HLAY,4H ENT,4HER T,4HHE,4HHRHO,4H COC
X,4HRDIN,4HATE,4HASSO,4HCIA T,4HITH,4HA TH,4HETA
X,4HOF,4HZERO,4H,OF,4H A P,4HOINT,4H NEA,4HR TH,4HE RO
X,4HOT,4HTO B,4HE DE,4HLETE,4HD. E,4HNTER,4H THE,4H VAL
X,4HUE,4HWITH,4HIN T,4HHE B,4HOX P,4HROVI,4HDED,4H INC
X,4HLUDE,4HA DE,4HCIMA,4HL, A,4HND M,4HINUS,4H SIG,4HN AS
X,4HAPPR,4HOPRI,4HATE.,4H NO,4HSIGN,4H IS,4HPOSI/
DATA LTPDRZ(64)/4H+IVE/
CALL CHCKSZ(-1,1)
IF(IERROR.NE.0)RETURN
CALL ERASE
CALL MOVABS(0,528)
DO 30 I=1,57,8
CALL ANMODE
WRITE(6,100)(LTPDRZ(I+J-1),J=1,8)
CALL RECOVER
CALL NEWLIN
100 FORMAT(21X,8A4)
CALL NEWLIN
CALL ANMODE
WRITE(6,101)
101 FORMAT(27X,'RHO.')
```



```

103 CALL RECOVER
    CALL NEWLIN (320,302)
    CALL MOVABS (0,22)
    CALL DRWREL (138,0)
    CALL DRWREL (0,-22)
    CALL DRWREL (-138,0)
    CALL MOVREL (0,3)
    CALL CARTN
    CALL ANMODE
    WRITE (6,103)
    FORMAT (22X,'>')
    READ (5,104) RHO
    FORMAT (F10.7,5X,F10.7)
104 CALL RECOVER
    CALL LOCATE (1,RHO,0.,LINEPZ)
    CALL NEWLIN
    CALL NEWLIN
    CALL ANMODE
    THETA=0.
    WRITE (6,105) POLZRO(LINEPZ,3),POLZRO(LINEPZ,4)
    FORMAT (I2X,26HDELETED REAL ZERO WILL BE ,F10.7,3X,F10.7)
105 CALL RECOVER
    CALL ANGLE (677,260)
    CALL NEWLIN
    CALL ANMODE
    WRITE (6,106)
    FORMAT (21X,32HIF CORRECT TYPE 'Y', IF NOT 'N'.)
106 CALL RECOVER
    CALL SCURSR (ICAR,IX,IY)
    IF (ICAR.NE.121.AND.(ICAR.NE.89)GO TO 10
    CALL UPDATE (1,LINEPZ)
    CALL NUC
    RETURN
END

SUBROUTINE DRPPLR
COMMON IXXXX(4),POLZRO(20,5),IERROR
DIMENSION LTPDRP(64)
DATA LTPDRP/4HTO D,4HELET,4HE A ,4HREAL,4H POL,4HE FR,4HOM T
,4HHE
,4HUNIT,4H CIR,4HCLE,4H ENT,4HER T,4HHE ,4HHRHO,
,4HCOOR,4HDINA,4HTE F,4HOR A,4H TH,4HETA,4H VAL
,4HOF Z,4HERO,4H OF ,4HA PO,4HINT ,4HNEAR,4H THE
,4HROOT,4H TO ,4HBE D,4HELET,4HED.,4HENTE,4HHR TH
,4HVALU,4HE WI,4HTHIN,4H THE,4H BOX,4H PRO,4HVIDE
,4HINCL,4HUDE ,4HA DE,4HCIMA,4HL , A,4HND M,4HINUS
,4H
,4HSIGN,4H AS ,4HAPPR,4HOPRI,4HATE.,4H
,4H
DATA LTPDRP(64)/4H

```





```

10  CALL CHCKSZ(-1,3)
    IF(IEOR.NE.0) RETURN
    CALL ERASE
    CALL MOVABS(0,528)
    DO 30 I=1,57,8
    CALL ANMODE
    WRITE(6,100)(LTPDRP(I+J-1),J=1,8)
    CALL RECOVR
    CALL NEWLIN
    FORMAT(21X,8A4)
    CALL NEWLIN
    CALL ANMODE
    WRITE(6,101)
    FORMAT(27X,'RHO')
    CALL RECOVR
    CALL NEWLIN
    CALL MOVABS(320,302)
    CALL DRWREL(0,22)
    CALL DRWREL(138,0)
    CALL DRWREL(0,-22)
    CALL DRWREL(-138,0)
    CALL MOVREL(0,3)
    CALL CARTN
    CALL ANMODE
    WRITE(6,103)
    FORMAT(22X,'>')
    READ(5,104) RHO
    FORMAT(F10.7)
    CALL RECOVR
    CALL NEWLIN
    CALL LOCATE(3,RHO,0.,LINEPZ)
    CALL NEWLIN
    CALL ANMODE
    THETA=0.
    WRITE(6,105)POLZRO(LINEPZ,3),POLZRO(LINEPZ,4)
    FORMAT(12X,26HDELETED REAL POLE WILL BE 'F10.7,3X,F10.7)
    CALL RECOVR
    CALL ANGLE(677,260)
    CALL NEWLIN
    CALL ANMODE
    WRITE(6,106)
    FORMAT(21X,32HIF CORRECT TYPE 'Y', IF NOT 'N'.)
    CALL RECOVR
    CALL SCURSR(ICAR,IX,IY)
    IF(ICAR.NE.121.AND.ICAR.NE.89)GO TO 10
    CALL UPDATE(3,LINEPZ)
    CALL NUC
    RETURN

```

ASDF4537  
ASDF4538  
ASDF4539  
ASDF4540  
ASDF4541  
ASDF4542  
ASDF4543  
ASDF4544  
ASDF4545  
ASDF4546  
ASDF4547  
ASDF4548  
ASDF4549  
ASDF4550  
ASDF4551  
ASDF4552  
ASDF4553  
ASDF4554  
ASDF4555  
ASDF4556  
ASDF4557  
ASDF4558  
ASDF4559  
ASDF4560  
ASDF4561  
ASDF4562  
ASDF4563  
ASDF4564  
ASDF4565  
ASDF4566  
ASDF4567  
ASDF4568  
ASDF4569  
ASDF4570  
ASDF4571  
ASDF4572  
ASDF4573  
ASDF4574  
ASDF4575  
ASDF4576  
ASDF4577  
ASDF4578  
ASDF4579  
ASDF4580  
ASDF4581  
ASDF4582  
ASDF4583  
ASDF4584





ASDF4585  
ASDF4586  
ASDF4587  
ASDF4588  
ASDF4589  
ASDF4590  
ASDF4591  
ASDF4592  
ASDF4593  
ASDF4594  
ASDF4595  
ASDF4596  
ASDF4597  
ASDF4598  
ASDF4599  
ASDF4600  
ASDF4601  
ASDF4602  
ASDF4603  
ASDF4604  
ASDF4605  
ASDF4606  
ASDF4607  
ASDF4608  
ASDF4609  
ASDF4610  
ASDF4611  
ASDF4612  
ASDF4613  
ASDF4614  
ASDF4615  
ASDF4616  
ASDF4617  
ASDF4618  
ASDF4619  
ASDF4620  
ASDF4621  
ASDF4622  
ASDF4623  
ASDF4624  
ASDF4625  
ASDF4626  
ASDF4627  
ASDF4628  
ASDF4629  
ASDF4630  
ASDF4631  
ASDF4632

```

END
SUBROUTINE ANGLE(IX,IY)
CALL MOVABS(IX,IY)
CALL MOVREL(14,20)
CALL DRWREL(-9,-18)
CALL DRWREL(20,0)
CALL DRWREL(-3,-1)
CALL DRWREL(-1,7)
CALL DRWREL(-3,5)
CALL DRWREL(-3,3)
CALL DRWREL(-4,2)
CALL DRWREL(0,-20)
RETURN
END

SUBROUTINE ERROR
COMMON IROOTS(4),POLZRO(20,5), IERROR
CALL LST
CALL MOVABS(0,154)
CALL ANMODE
GO TO (2,2,2,4,4,4,6,8,8,6,8,6,6),IERROR
WRITE(6,100)
FORMAT(21X,'MAXIMUM SYSTEM ORDER EXCEEDED.')
GO TO 10
WRITE(6,101)
FORMAT(19X,'ROOTS NOT AVAILABLE TO BE DELETED.')
GO TO 10
WRITE(6,102)
FORMAT(11X,'POLES MAY NOT BE ENTERED OUTSIDE THE UNIT CIRCLE.')
GO TO 10
WRITE(6,103)
FORMAT(10X,'ZEROS MUST BE WITHIN TEN UNITS OF THE UNIT CIRCLE.')
CALL RECOVER
CALL NEWLIN
CALL ANMODE
WRITE(6,104)
FORMAT(25X,'COMMAND NOT PROCESSED.')
CALL RECOVER
CALL NEWLIN
CALL ANMODE
WRITE(6,105)
FORMAT(11X,'WHEN YOU ARE READY TO RETURN HIT',
1, ANY KEYBOARD KEY')
CALL RECOVER
CALL SCURSR(ICAR, IX, IY)
IERROR=0
CALL NUC

```



RETURN  
END

SUBROUTINE LSI  
COMMON IROOTS(4), POLZRO(20,5), IERROR  
DIMENSION IRTIP(5)  
DATA IZ/1HZ/, IP/1HP/, IZP/2HZP/  
DATA IRTIP/2H , 2HRZ, 2HCZ, 2HRP, 2HCP/  
CALL ERASE  
CALL HOME  
CALL ANMODE

WRITE(6,102) (IZ, I=1, 70)  
FORMAT(1X, 70A1)

102

WRITE(6,108)  
FORMAT(1X, Z', 5X, 'REAL', 6X, Z', 3X, 'IMAGINARY', 3X, Z',  
1, 6X, 'RHO', 6X, Z', 5X, 'THETA', 5X, Z RT Z')

108

CALL NEWLIN  
CALL RECOVER  
CALL ANMODE  
WRITE(6,102) (IZ, I= 1, 70)  
CALL RECOVER  
CALL NEWLIN  
DO 10 I=1, 10  
CALL ANMODE

ITYPE=POLZRO(I, 5)+1.  
IF(POLZRO(I, 5).NE.2..AND.POLZRO(I, 5).NE.4.)  
1WRITE(6,104) (POLZRO(I, J), J=1, 4), IRTIP(ITYPE)  
IF(POLZRO(I, 5).EQ.2..OR.POLZRO(I, 5).EQ.4.)  
1WRITE(6,103) (POLZRO(I, J), J=1, 4), IRTIP(ITYPE)  
FORMAT(1, Z', F10.7, Z', F10.7, Z', F10.7,  
13X, Z', F10.7, Z', F10.7, Z', F10.7, Z', F10.7,  
1, Z', F10.7, Z', F10.7, Z', F10.7, Z', F10.7,  
CALL RECOVER  
CALL NEWLIN  
CONTINUE  
CALL ANMODE  
WRITE(6,112) (IZP, I=1, 35)  
FORMAT(1X, 35A2)

103

104

10

112

CALL NEWLIN  
DO 20 I=1, 20  
CALL ANMODE  
ITYPE=POLZRO(I, 5)+1.  
IF (POLZRO(I, 5).NE.2..AND.POLZRO(I, 5).NE.4.)

ASDF4633  
ASDF4634  
ASDF4635  
ASDF4636  
ASDF4637  
ASDF4638  
ASDF4639  
ASDF4640  
ASDF4641  
ASDF4642  
ASDF4643  
ASDF4644  
ASDF4645  
ASDF4646  
ASDF4647  
ASDF4648  
ASDF4649  
ASDF4650  
ASDF4651  
ASDF4652  
ASDF4653  
ASDF4654  
ASDF4655  
ASDF4656  
ASDF4657  
ASDF4658  
ASDF4659  
ASDF4660  
ASDF4661  
ASDF4662  
ASDF4663  
ASDF4664  
ASDF4665  
ASDF4666  
ASDF4667  
ASDF4668  
ASDF4669  
ASDF4670  
ASDF4671  
ASDF4672  
ASDF4673  
ASDF4674  
ASDF4675  
ASDF4676  
ASDF4677  
ASDF4678  
ASDF4679  
ASDF4680



```

1WRITE(6,106) (POLZRO(I,J), J=1,4), IRTIP(I,TYPEP)
  IF(POLZRO(I,5).EQ.2.OR.POLZRO(I,5).EQ.4.)
1WRITE(6,105) (POLZRO(I,J), J=1,4), IRTIP(I,TYPEP)
105  FORMAT(,P, F10.7, ,P, +/ -, F10.7, ,P, , F10.7,
13X, ,P, +/ -, F10.7, ,P, ,A2, ,P, , F10.7, ,P, , F10.7,
106 1, ,P, F10.7, ,P, ,A2, ,P, )
  CALL RECOVER
  CALL NEWLIN
  CONTINUE
  CALL ANMODE
  WRITE(6,102) (IP, I =1,70)
  CALL RECOVER
  CALL NEWLIN
  CALL ANMODE
  WRITE(6,107)
  FORMAT(//)
107  IF(IERROR.NE.0)RETURN
  CALL RECOVER
  CALL NEWLIN
  CALL ANMODE
  WRITE(6,110) IROOTS(1)
  FORMAT(20X, ,NUMBER OF REAL ZEROS:',I3)
110  CALL RECOVER
  CALL NEWLIN
  CALL ANMODE
  WRITE(6,111) IROOTS(2)
  FORMAT(20X, ,NUMBER OF COMPLEX ZERO PAIRS:',I3)
111  CALL RECOVER
  CALL NEWLIN
  CALL ANMODE
  WRITE(6,113) IROOTS(3)
  FORMAT(20X, ,NUMBER OF REAL POLES:',I3)
113  CALL RECOVER
  CALL NEWLIN
  CALL ANMODE
  WRITE(6,114) IROOTS(4)
  FORMAT(20X, ,NUMBER OF COMPLEX POLE PAIRS:',I3)
114  CALL RECOVER
  CALL NEWLIN
  CALL ANMODE
  WRITE(6,115)
  FORMAT(, HIT SPACE THEN RETURN')
115  CALL RECOVER
  CALL NEWLIN
  CALL ANMODE
  READ(5,116) IN
116  FORMAT(A1)

```

```

ASDF4681
ASDF4682
ASDF4683
ASDF4684
ASDF4685
ASDF4686
ASDF4687
ASDF4688
ASDF4689
ASDF4690
ASDF4691
ASDF4692
ASDF4693
ASDF4694
ASDF4695
ASDF4696
ASDF4697
ASDF4698
ASDF4699
ASDF4700
ASDF4701
ASDF4702
ASDF4703
ASDF4704
ASDF4705
ASDF4706
ASDF4707
ASDF4708
ASDF4709
ASDF4710
ASDF4711
ASDF4712
ASDF4713
ASDF4714
ASDF4715
ASDF4716
ASDF4717
ASDF4718
ASDF4719
ASDF4720
ASDF4721
ASDF4722
ASDF4723
ASDF4724
ASDF4725
ASDF4726
ASDF4727
ASDF4728

```





```
CALL RECOVER
CALL NUC
RETURN
END
```

```
SUBROUTINE PLTTBL
```

```
C* SUBROUTINE PLOTS THE POLZRO TABLE OF THE UNIT CIRCLE (12)
```

```
C* COMMON IXXXXX(4), POLZRO(20,5)
C* DO 10 I=1,20
```

```
C* IF POLZRO(I,5)=0 NO ROOT HAS BEEN STORED
```

```
C* IF (POLZRO(I,5).EQ.0.)GO TO 10
C* CALL PLOTPT(IFIX(POLZRO(I,5)),POLZRO(I,1),POLZRO(I,2))
C* CONTINUE
C* RETURN
C* END
```

```
10
```

```
SUBROUTINE CHCKSZ (IDELTA, ITYPE)
```

```
C* THIS SUBROUTINE CHECKS THE TABLE OF ROOTS TO SEE
C* IF IT IS POSSIBLE TO EXECUTE THE DESIRED SUBROUTINE
C* IDELTA: IS THE CHANGE IN THE NUMBER OF ROOTS EXPECTED
C* BY THE CALLING SUBROUTINE
C* ITYPE: IS THE TYPE OF ROOTS TO BE ALTERED, I.E.,
C* COMPLEX, REAL, ZERO, OR POLE
C* IERROR: RETURNS THE ERROR MESSAGE NUMBER IF IT IS
C* NOT POSSIBLE TO EXECUTE THE SELECTED COMMAND;
C* THE MAXIMUM SYSTEM ORDER IS LIMITED TO TEN AND
C* ROOTS MAY NOT BE DELETED IF THEY HAVE NOT BEEN INPUT
```

```
COMMON IROOTS(4),PXXXXX(20,5),IERROR
```

```
C* IROOTS(1) = FLAG COUNTING REAL ZEROS
C* IROOTS(2) = FLAG COUNTING COMPLEX ZERO PAIRS
C* IROOTS(3) = FLAG COUNTING REAL POLES
C* IROOTS(4) = FLAG COUNTING COMPLEX POLE PAIRS
```

```
C* DETERMINE WHETHER ROOT(S) ARE BEING ADDED OR DELETED (1)
```

```
IF(IDELTA.LT.0)GO TO 10
```

```
C* ROOT(S) ARE BEING ADDED (3)
C* CALCULATE THE EXPECTED ORDER OF THE SYSTEM (1)
```

```
ASDF4729
ASDF4730
ASDF4731
ASDF4732
ASDF4733
ASDF4734
ASDF4735
ASDF4736
ASDF4737
ASDF4738
ASDF4739
ASDF4740
ASDF4741
ASDF4742
ASDF4743
ASDF4744
ASDF4745
ASDF4746
ASDF4747
ASDF4748
ASDF4749
ASDF4750
ASDF4751
ASDF4752
ASDF4753
ASDF4754
ASDF4755
ASDF4756
ASDF4757
ASDF4758
ASDF4759
ASDF4760
ASDF4761
ASDF4762
ASDF4763
ASDF4764
ASDF4765
ASDF4766
ASDF4767
ASDF4768
ASDF4769
ASDF4770
ASDF4771
ASDF4772
ASDF4773
ASDF4774
ASDF4775
ASDF4776
```





```

C* IF (ITYPE.EQ.1.OR.ITYPE.EQ.2)NORDER=IROOTS(1)+2*IROOTS(2)+IDELTA
C* IF (ITYPE.EQ.3.OR.ITYPE.EQ.4)NORDER=IROOTS(3)+2*IROOTS(4)+IDELTA
C* IF "NORDER" IS GREATER THAN 10, MAXIMUM SYSTEM ORDER HAS BEEN
C* EXCEEDED AND AN ERROR RESULTS (1)
C*
20 IF(NORDER.GT.10)IERROR=ITYPE
C* IF(Y.GT.0.)CALL POINTA(XX,YY)
C* RETURN
C*
C* ROOTS ARE BEING DELETED (2)
C* CALCULATE THE EXPECTED ROOT TOTAL IF THE CALLING
C* SUBROUTINE EXECUTES (1)
C* PLUS SIGN IS DUE TO "IDELTA" BEING LESS THAN ZERO
C*
10 IXPTOT=IROOTS(ITYPE)+IDELTA
C*
C* IF THE EXPECTED ROOTS TOTAL "IXPTOT" IS NEGATIVE
C* AN ERROR IS GENERATED (1)
C*
999 IF(IXPTOT.LT.0)IERROR=ITYPE+4
C* RETURN
C* END
C*
C* SUBROUTINE UPDATE(ITYPE,LINEPZ)
C*
C* THIS SUBROUTINE WILL UPDATE THE POLZRO TABLE WHEN A ROOT IS REMOVED
C*
C* COMMON IROOTS(4),POLZRO(20,5)
C*
C* "LINEPZ" IS THE LINE OF POLZRO ARRAY TO BE DELETED
C*
10 IF(LINEPZ.EQ.20)GO TO 45
15 IF(LINEPZ.GT.10)GO TO 30
20 IF(LINEPZ.EQ.10)GO TO 15
C* DO 10 J=LINEPZ,9
C* DO 10 K=1,5
C* POLZRO(J,K)=POLZRO(J+1,K)
C* DO 20 M=1,4
C* POLZRO(10,M)=-10.
C* POLZRO(10,5)=0.
C* GO TO 60
30 DO 40 J=LINEPZ,19
C* DO 40 K=1,5
C* POLZRO(J,K)=POLZRO(J+1,K)
40 DO 50 M=1,4
45 POLZRO(20,M)=-10.
50 POLZRO(20,5)=0.

```



```

C* ADJUST THE ROOT COUNTER SINCE A ROOT HAS BEEN DELETED
C*
C* 60 IROOTS(ITYPE)=IROOTS(ITYPE)-1
C* RETURN
C* END
C* SUBROUTINE LOCATE(ITYPE,X,Y,LINEPZ)
C* THIS SUBROUTINE LOCATES THE STORED ROOT NEAREST THE CURSOR POSITION
C*
C* COMMON IXXXX(4), POLZRO(20,5)
C*
C* "ITYPE" IS THE TYPE OF ROOT SEARCHED
C* "LINEPZ" IS THE LINE OF POLZRO WHICH CONTAINS THE ROOT
C* OF THE DESIRED TYPE WHICH IS CLOSEST TO THE
C* USER-POSITIONED CURSOR
C* "DIST" IS THE SHORTEST DISTANCE FOUND FROM STORED ROOT
C* TO THE USER-POSITIONED CURSOR
C*
C* STRRTS STORES ONLY THE COMPLEX POLE IN THE TOP HALF PLANE
C*
C* SY=ABS(Y)
C* DIST=20.
C* DO 10 I=1,20
C* IF( IFIX(POLZRO(I,5)).NE.ITYPE)GO TO 10
C* CKDIST=SQRT((X-POLZRO(I,1))**2+(Y-POLZRO(I,2))**2)
C* IF(CKDIST.LT.DIST)LINEPZ=I
C* IF(CKDIST.LT.DIST)DIST=CKDIST
C* CONTINUE
C* RETURN
C* END
C* 10
C* SUBROUTINE START
C* THIS SUBROUTINE INITIALIZES ALL THE VARIABLES STORED IN
C* COMMON BEFORE EXECUTION OF THE PROGRAM (10)
C*
C* COMMON IROOTS(4), POLZRO(20,5), IERROR, IOLD
C* IOLD=0
C* DO 10 I=1,4
C* IROOTS(I)=0
C* IERROR=0
C* DO 20 J=1,20
C* POLZRO(J,5)=0.
C* DO 20 K=1,4
C* POLZRO(J,K)=-10.
C* RETURN
C* 10
C* 20

```

ASDF4825  
ASDF4826  
ASDF4827  
ASDF4828  
ASDF4829  
ASDF4830  
ASDF4831  
ASDF4832  
ASDF4833  
ASDF4834  
ASDF4835  
ASDF4836  
ASDF4837  
ASDF4838  
ASDF4839  
ASDF4840  
ASDF4841  
ASDF4842  
ASDF4843  
ASDF4844  
ASDF4845  
ASDF4846  
ASDF4847  
ASDF4848  
ASDF4849  
ASDF4850  
ASDF4851  
ASDF4852  
ASDF4853  
ASDF4854  
ASDF4855  
ASDF4856  
ASDF4857  
ASDF4858  
ASDF4859  
ASDF4860  
ASDF4861  
ASDF4862  
ASDF4863  
ASDF4864  
ASDF4865  
ASDF4866  
ASDF4867  
ASDF4868  
ASDF4869  
ASDF4870  
ASDF4871  
ASDF4872



```

END
SUBROUTINE STRRTS(ITYPE,MODE,ARGMT1,ARGMT2)
C* THIS SUBROUTINE IS DESIGNED TO STORE ROOTS PREVIOUSLY
C* GENERATED IN THE ROOT TABLE "POLZRO"
C*
COMMON IROOTS(4), POLZRO(20,5), IERROR
PI=3.14159265
IF(ARGMT1.EQ.0..AND.ARGMT2.EQ.0.)GO TO 45
C* GIVEN "MODE", "ARGMT1", AND "ARGMT2", GENERATE THE REAL,
C* IMAGINARY, RADIUS, AND THETA VALUES OF THE UPPER HALF PLANE
C* ROOT TO BE STORED (THIS INCLUDES THE REAL AXIS).
C*
GO TO (10,20),MODE
C* "MODE" DICTATES THAT "ARGMT1" AND "ARGMT2" ARE RECTANGULAR
C*
10 RREAL=ARGMT1
RIMAG=ABS(ARGMT2)
RADIUS=SQRT(ARGMT1**2+ARGMT2**2)
THETA=ABS(ATAN2(ARGMT1,ARGMT2))
GO TO 30
C* "MODE" DICTATES THAT "ARGMT1" AND "ARGMT2" ARE POLAR
C*
20 RADIUS=ARGMT1
IF(ARGMT2.LE.PI..AND.ARGMT2.GT.-PI)GO TO 40
IF(ARGMT2.LE.-PI)ARGMT2=ARGMT2+2.*PI
IF(ARGMT2.GT.PI)ARGMT2=ARGMT2-2.*PI
GO TO 50
30 THETA=ARGMT2
RREAL=RADIUS*COS(THETA)
RIMAG=RADIUS*SIN(THETA)
GO TO 30
40
C* IF THE ROOT IS TO BE AT THE ORIGIN,
C* ASSIGN VALUES SUCH THAT ILLEGAL LIBRARY FUNCTION
C* ARGUMENTS WILL NOT BE ASSIGNED
C*
45 RREAL=0.
RIMAG=0.
RADIUS=0.
THETA=0.
C* STORAGE PROCEDURE IS A FUNCTION OF THE TYPE OF ROOTS TO BE STORED
C*

```





```

C* 30      RZ  CZ  RP  CP
C*      GO TO (110,210,120,220), ITYPE
C*
C* A REAL ZERO IS BEING STORED
C* IROOTS(1) IS THE NUMBER OF REAL ZEROS ALREADY STORED
C*
110  POLZRO(IROOTS(1)+IROOTS(2)+1,1)=RREAL
    POLZRO(IROOTS(1)+IROOTS(2)+1,2)=0.
    POLZRO(IROOTS(1)+IROOTS(2)+1,3)=ABS(RREAL)
    IF(ARGMT1.GE.0.)THETA=0.
    IF(ARGMT1.LT.0.)THETA=PI
    POLZRO(IROOTS(1)+IROOTS(2)+1,4)=THETA
    POLZRO(IROOTS(1)+IROOTS(2)+1,5)=1.
    IROOTS(1)=IROOTS(1)+1
    GO TO 999
C*
C* A REAL POLE IS TO BE STORED
C* IROOTS(3) IS THE NUMBER OF REAL POLES ALREADY STORED
C*
120  POLZRO(IROOTS(3)+IROOTS(4)+1,1)=RREAL
    POLZRO(IROOTS(3)+IROOTS(4)+1,2)=0.
    POLZRO(IROOTS(3)+IROOTS(4)+1,3)=ABS(RREAL)
    IF(ARGMT1.GE.0.)THETA=0.
    IF(ARGMT1.LT.0.)THETA=PI
    POLZRO(IROOTS(3)+IROOTS(4)+1,4)=THETA
    POLZRO(IROOTS(3)+IROOTS(4)+1,5)=3.
    IROOTS(3)=IROOTS(3)+1
    GO TO 999
C*
C* A COMPLEX ZERO IS TO BE STORED
C* IROOTS(2) IS THE NUMBER OF COMPLEX ZERO PAIRS ALREADY STORED
C*
210  POLZRO(IROOTS(2)+IROOTS(1)+1,1)=RREAL
    POLZRO(IROOTS(2)+IROOTS(1)+1,2)=ABS(RIMAG)
    POLZRO(IROOTS(2)+IROOTS(1)+1,3)=ABS(RADIUS)
    IF(ARGMT1.EQ.0.)AND.ARGMT2.EQ.0.)RADIUS=1.
    POLZRO(IROOTS(2)+IROOTS(1)+1,4)=ARCOS(RREAL/ABS(RADIUS))
    POLZRO(IROOTS(2)+IROOTS(1)+1,5)=2.
    IROOTS(2)=IROOTS(2)+1
    GO TO 999
C*
C* A COMPLEX POLE IS TO BE STORED
C* IROOTS(4) IS THE NUMBER OF COMPLEX POLE PAIRS ALREADY STORED
C*
220  POLZRO(IROOTS(4)+IROOTS(3)+1,1)=RREAL
    POLZRO(IROOTS(4)+IROOTS(3)+1,2)=RIMAG
    POLZRO(IROOTS(4)+IROOTS(3)+1,3)=ABS(RADIUS)
    IF(ARGMT1.EQ.0.)AND.ARGMT2.EQ.0.)RADIUS=1.

```





```

999 POLZRO(IROOTS(4)+IROOTS(3)+11,4)=ARCOS(RREAL/ABS(RADIUS))
POLZRO(IROOTS(4)+IROOTS(3)+11,5)=4.
IROOTS(4)=IROOTS(4)+1
IF(ITYPE.EQ.1.OR.ITYPE.EQ.3)CALL PLOTRT(ITYPE,RREAL,0.)
IF(ITYPE.EQ.2.OR.ITYPE.EQ.4)CALL PLOTRT(ITYPE,RREAL,RIMAG)
RETURN
END

SUBROUTINE PLOTRT (ITYPE, RREAL, RIMAG)
C* THE PURPOSE OF THIS SUBROUTINE IS TO PLOT
C* THE ROOT JUST ENTERED INTO THE SYSTEM
C*
C* DIMENSION IZERO(16), ISTAR(16)
C*
C* DATA POINTS REQUIRED FOR DRAWING ZEROS
C*
C* DATA IZERO/2,-2,0,-2,-2,-2,-2,0,-2,2,0,2,2,2,2,0/
C* DATA ISTAR/-5,5,10,-10,-5,10,0,-10,5,10,-10,-10,5,-10,0/
C* IF ZERO IS OUTSIDE THE UNIT CIRCLE, PLOT A STAR.
C*
C* RADIUS = SQRT(RREAL**2 + RIMAG**2)
C* IF(RADIUS.GT.1.)GO TO 300
C* SINCE ROOT IS INSIDE THE UNIT CIRCLE
C*
C* IF THE ROOT IS A ZERO GO TO 100
C* IF THE ROOT IS A POLE GO TO 200
C*
C* RZ CZ RP CP
C* GO TO (100,100,200,200),ITYPE
C*
C* CENTER THE ZERO ON THE LOCATION STORED IN POLZRO
C*
C* 100 CALL MOVEA(RREAL,RIMAG)
C* GENERATE THE ZERO (3)
C*
C* CALL MOVREL(1,3)
C* DO 120 I=1,15,2
C* 120 CALL DRWREL(IZERO(I),IZERO(I+1))
C* IF THE ZERO IS REAL, RETURN
C*
C* IF(ITYPE.EQ.1)GO TO 999
C* PLOT THE CORRESPONDING COMPLEX ZERO IN THE LOWER HALF PLANE (4)
C*

```

```

ASDF4969
ASDF4970
ASDF4971
ASDF4972
ASDF4973
ASDF4974
ASDF4975
ASDF4976
ASDF4977
ASDF4978
ASDF4979
ASDF4980
ASDF4981
ASDF4982
ASDF4983
ASDF4984
ASDF4985
ASDF4986
ASDF4987
ASDF4988
ASDF4989
ASDF4990
ASDF4991
ASDF4992
ASDF4993
ASDF4994
ASDF4995
ASDF4996
ASDF4997
ASDF4998
ASDF4999
ASDF5000
ASDF5001
ASDF5002
ASDF5003
ASDF5004
ASDF5005
ASDF5006
ASDF5007
ASDF5008
ASDF5009
ASDF5010
ASDF5011
ASDF5012
ASDF5013
ASDF5014
ASDF5015
ASDF5016

```



```

CALL MOVEA(RREAL,-RIMAG)
CALL MOVREL(1,3)
DO 130 I=1,15,2
130 CALL DRWREL (IZERO(I), IZERO(I+1))
GO TO 999

C* PLOT THE POLE (REAL, OR UPPER HALF PLANE) (5)
C*
C* 200 CALL MOVEA (RREAL,RIMAG)
CALL MOVREL(3,3)
CALL DRWREL(-6,-6)
CALL MOVREL(0,5)
CALL DRWREL(6,-6)

C* IF THE POLE IS REAL, RETURN
C*
C* IF(ITYPE.EQ.3)GO TO 999

C* IF THE POLE IS COMPLEX, PLOT THE CORRESPONDING
C* LOWER HALF PLANE POLE (5)
C*
CALL MOVEA (RREAL,-RIMAG)
CALL MOVREL(3,3)
CALL DRWREL(-6,-6)
CALL MOVREL(0,6)
CALL DRWREL(6,-6)
GO TO 999

300 ANGLE=ATAN2(RIMAG,RREAL)
PREAL=1.05*COS(ANGLE)
PIMAG=1.05*SIN(ANGLE)
CALL MOVEA(PREAL,PIMAG)
DO 320 M=1,2
DO 310 I=1,16,4
CALL MOVREL(ISTAR(I), ISTAR(I+1))
CALL DRWREL(ISTAR(I+2), ISTAR(I+3))
CONTINUE
310 IF(M.EQ.2.OR.ITYPE.EQ.1)GO TO 999
320 CALL MOVEA(PREAL,-PIMAG)
999 RETURN
END
ASDF5017
ASDF5018
ASDF5019
ASDF5020
ASDF5021
ASDF5022
ASDF5023
ASDF5024
ASDF5025
ASDF5026
ASDF5027
ASDF5028
ASDF5029
ASDF5030
ASDF5031
ASDF5032
ASDF5033
ASDF5034
ASDF5035
ASDF5036
ASDF5037
ASDF5038
ASDF5039
ASDF5040
ASDF5041
ASDF5042
ASDF5043
ASDF5044
ASDF5045
ASDF5046
ASDF5047
ASDF5048
ASDF5049
ASDF5050
ASDF5051
ASDF5052
ASDF5053
ASDF5054
ASDF5055
ASDF5056

```



## APPENDIX H: SOURCE DECK FOR ASDP COMMAND: RESPONSE

```

C** SOURCE DECK FOR ASDF COMMAND: RESPONSE **
C** **** * ***** **
C** ***** POLZRO(20,5), IERROR *****
C** COMMON IROOTS(4), POLZRO(20,5), IERROR *****
C** COMMON /ABVCTR/ A(11), B(11) *****
C** COMMON /IMPULS/ YI(1025), YS(1025), EN(1025) *****
C** COMMON /FREQ/ YP(1026), YM(1026), PI(1026) *****
C** LINEAR=1 *****
C** LOG=2 *****
C**
C** INITIALIZE A AND B TO ZERO
C** DO 5 K=1,20
C** A(K)=0.
C** B(K)=0.
C**
C** READ THE POLZRO TABLE OFF OF THE FILE
C** DO 20 I=1,20
C** FORMAT(IX,5F12.8)
C** READ(1,100) (POLZRO(I,J),J=1,5)
C**
C** READ THE IROOTS VALUES FROM THE FILE
C** READ(1,101) (IROOTS(I),I=1,4)
C** FORMAT(IX,4I2)
C** READ(1,102) FLTRGN
C** FORMAT(IX,F12.8)
C** FLTRGN=ABS(FLTRGN)
C** ICHAR=0
C**
C** PRINT OUT A TABLE OF THE POLES AND ZEROS
C** CALL FLTR2(FLTRGN)
C**
C** GENERATE THE A(K) COEFFICIENTS
C** CALL ASUBK(A,VORDER)
C** NDEN=NORDER+1
C**
C** GENERATE THE B(R) COEFFICIENTS
C** CALL BSUBR(B,VORDER)
C** NNUM=NORDER+1
C**

```





```

C* CHECK TO SEE IF THE FILTER IS CAUSAL
C*      IF(NNUM.GT.NDEN)WRITE(6,103)
C*      103      FORMAT(10('/'),' FILTER IS NOT CAUSAL - PROGRAM TERMINATES')
C*      IF(NNUM.GT.NDEN)STOP
C* ADJUST THE COEFFICIENTS IF NNUM DOES NOT EQUAL NDEN
C*      IF(NNUM.LT.NDEN)CALL ADJUST(NNUM,NDEN)
C* COMPUTE THE UNIT SAMPLE RESPONSE
C*      CALL RSPNSE(0,FLTRGN)
C* PLOT THE UNIT SAMPLE RESPONSE
C*      CALL PLTSTP(1,FLTRGN)
C*      CALL PLTSTP(1,FLTRGN)
C*      IF(ICHAR.EQ.88.OR.ICHAR.EQ.120)GO TO 30
C* CALCULATE THE STEP RESPONSE
C*      CALL RSPNSE(1,FLTRGN)
C* PLOT THE STEP RESPONSE
C*      CALL PLTSTP(1,FLTRGN)
C*      IF(ICHAR.EQ.88.OR.ICHAR.EQ.120)GO TO 30
C* COMPUTE THE MAGNITUDE AND PHASE OF THE TRANSFER FUNCTION
C*      CALL FRQNCY(NNUM,NDEN,FLTRGN)
C* PLOT THE PHASE OF THE TRANSFER FUNCTION
C*      CALL PLTTRF(3,LINEAR,ICHAR)
C*      IF(ICHAR.EQ.88.OR.ICHAR.EQ.120)GO TO 30
C* PLOT THE MAGNITUDE OF THE TRANSFER FUNCTION
C*      FIRST WITH LINEAR AXIS, THEN IN DECIBELS
C*      CALL PLTTRF(4,LINEAR,ICHAR)
C*      IF(ICHAR.EQ.88.OR.ICHAR.EQ.120)GO TO 30
C*      CALL PLTTRF(4,LOG,ICHAR)
C*      IF(ICHAR.EQ.88.OR.ICHAR.EQ.120)GO TO 30
C*      STOP
C*      END
C* SUBROUTINE ASUBK (POLE,NORDER)

```

```

ASDF6049
ASDF6050
ASDF6051
ASDF6052
ASDF6053
ASDF6054
ASDF6055
ASDF6056
ASDF6057
ASDF6058
ASDF6059
ASDF6060
ASDF6061
ASDF6062
ASDF6063
ASDF6064
ASDF6065
ASDF6066
ASDF6067
ASDF6068
ASDF6069
ASDF6070
ASDF6071
ASDF6072
ASDF6073
ASDF6074
ASDF6075
ASDF6076
ASDF6077
ASDF6078
ASDF6079
ASDF6080
ASDF6081
ASDF6082
ASDF6083
ASDF6084
ASDF6085
ASDF6086
ASDF6087
ASDF6088
ASDF6089
ASDF6090
ASDF6091
ASDF6092
ASDF6093
ASDF6094
ASDF6095
ASDF6096

```





```

C* GENERATE THE AK COEFFICIENTS FROM THE POLES
C*
COMMON IROOTS(4), POLZRO(20,5), IERROR
COMPLEX POLES(11), ACOEF(11)
REAL POLE(11)

C* INITIALIZE ALL THE VECTORS AND INDICES TO ZERO
C*
DO 5 K=1,11
POLES(K)=(0.,0.)
POLE(K)=0.
ACOE(K)=0.
5
C* "J" INDEXES THE TABLE "POLES"
C* "K" INDEXES THE TABLE "POLZRO"
C*
J=1
K=1
C* COMPUTE THE ORDER OF THE SYSTEM OF POLES
C*
NORDER=IROOTS(3)+2*IROOTS(4)
IF(NORDER.NE.0)GO TO 10
POLE(1)=1.0
RETURN
10
C* THE VECTOR "POLES" WILL CONTAIN A LIST OF ALL
C* THE POLES, IE., X+J0, X+JY, X-JY, . . .
C*
POLES(J)=CMPLX(POLZRO(10+K,1),POLZRO(10+K,2))
C*
TEST TO SEE IF ALL ROOTS HAVE BEEN
C* STORED IN THE "POLES" VECTOR
C*
IF(J.EQ.NORDER)GO TO 20
J=J+1
C*
IF THE ROOT IS REAL CONTINUE PROCESING THE NEXT ROOT
C*
IF(POLZRO(K+10,5).NE.3)GO TO 15
K=K+1
GO TO 10
C*
SINCE THE ROOT IS COMPLEX PUT THE CONJUGATE
C* OF THE ROOT IN THE "POLES" TABLE.
C*
15 POLES(J)=CMPLX(POLZRO(10+K,1),-POLZRO(10+K,2))
ASDF6097
ASDF6098
ASDF6099
ASDF6100
ASDF6101
ASDF6102
ASDF6103
ASDF6104
ASDF6105
ASDF6106
ASDF6107
ASDF6108
ASDF6109
ASDF6110
ASDF6111
ASDF6112
ASDF6113
ASDF6114
ASDF6115
ASDF6116
ASDF6117
ASDF6118
ASDF6119
ASDF6120
ASDF6121
ASDF6122
ASDF6123
ASDF6124
ASDF6125
ASDF6126
ASDF6127
ASDF6128
ASDF6129
ASDF6130
ASDF6131
ASDF6132
ASDF6133
ASDF6134
ASDF6135
ASDF6136
ASDF6137
ASDF6138
ASDF6139
ASDF6140
ASDF6141
ASDF6142
ASDF6143
ASDF6144

```



ASDF6145  
ASDF6146  
ASDF6147  
ASDF6148  
ASDF6149  
ASDF6150  
ASDF6151  
ASDF6152  
ASDF6153  
ASDF6154  
ASDF6155  
ASDF6156  
ASDF6157  
ASDF6158  
ASDF6159  
ASDF6160  
ASDF6161  
ASDF6162  
ASDF6163  
ASDF6164  
ASDF6165  
ASDF6166  
ASDF6167  
ASDF6168  
ASDF6169  
ASDF6170  
ASDF6171  
ASDF6172  
ASDF6173  
ASDF6174  
ASDF6175  
ASDF6176  
ASDF6177  
ASDF6178  
ASDF6179  
ASDF6180  
ASDF6181  
ASDF6182  
ASDF6183  
ASDF6184  
ASDF6185  
ASDF6186  
ASDF6187  
ASDF6188  
ASDF6189  
ASDF6190  
ASDF6191  
ASDF6192

```

IF(J.EQ.NORDER) GO TO 20
J=J+1
K=K+1
GO TO 10

C* GENERATE THE COEFFICIENTS FOR THE D(Z) POLYNOMIAL
C* OF THE TRANSFER FUNCTION H(Z).
C*
20 CALL COEFF(POLES,NORDER,ACOE)
C* CHANGE THE COEFFICIENTS FROM COMPLEX ASUBK TO REAL ASUBK
C*
40 DO 40 J=1,11
   POLE(J)=REAL(ACOE(J))
   RETURN
   END

SUBROUTINE BSJBR(ZERO,NORDER)
GENERATE THE B(R) COEFFICIENTS FROM THE ZEROS
COMMON IROOTS(4),POLZRO(20,5),IERROR
COMPLEX ZEROS(11),BCOE(11)
REAL ZERO(11)

C* INITIALIZE ALL VECTORS AND INDICES
C*
DO 5 K=1,11
  ZEROS(K)=(0.,0.)
  ZERO(K)=0.
  BCOE(K)=(0.,0.)
5
C* "J" INDEXES THE "ZEROS" TABLE
C* "K" INDEXES THE "POLZRO" TABLE
C*
J=1
K=1

C* COMPUTE THE ORDER OF THE SYSTEM OF ZEROS.
C*
NORDER=IROOTS(1)+2*IROOTS(2)
IF(NORDER.NE.0)GO TO 10
ZERO(1)=1.0
RETURN

C* THE VECTOR "ZEROS" CONTAINS A LIST OF ALL
C* OF THE ZEROS, IE., X+J0, X+JY, X-JY, ...
C*

```



```

10   ZEROS(J)=CMPLX(POLZRO(K,1),POLZRO(K,2))
C*   CHECK TO SEE IF ALL THE ROOTS HAVE BEEN PROCESSED
C*
C*   IF(J.EQ.NORDER)GO TO 20
C*   J=J+1
C*
C*   IF THE ROOT IS REAL CONTINUE PROCESSING THE NEXT ROOT
C*
C*   IF(POLZRO(K,5).NE.1)GO TO 15
C*   K=K+1
C*   GO TO 10
C*
C*   SINCE THE ROOT IS COMPLEX, ENTER THE CONJUGATE
C*   INTO THE TABLE "ZEROS".
C*
C*   15   ZEROS(J)=CMPLX(POLZRO(K,1),-POLZRO(K,2))
C*   IF(J.EQ.NORDER)GO TO 20
C*   J=J+1
C*   K=K+1
C*   GO TO 10
C*
C*   GENERATE THE BSUBR COEFFICIENTS
C*
C*   20   CALL COEFF(ZEROS, NORDER, BCOEF)
C*
C*   MAKE THE COEFFICIENTS BSUBR REAL.
C*
C*   40   DO 40 J=1,11
C*   ZERO(J)=REAL(BCOEF(J))
C*   RETURN
C*   END
C*
C*   SUBROUTINE COEFF(V,N,H1)
C*
C*   "V" - VECTOR OF ROOTS; "N" - THE ORDER OF THE SYSTEM
C*   "H1" - VECTOR WITH THE FINAL COEFFICIENTS
C*   H(1)=Z***(0), H(2)=Z***(-1), . . . ,H(11)=Z***(-10)
C*
C*   COMPLEX H1(11),H2(11),H3(11),V(11)
C*   INITIALIZE VECTORS
C*   IZERO=0
C*   DO 10 I=1,11
C*   H1(I)=(0.,0.)
C*   H2(I)=(0.,0.)
C*   H3(I)=(0.,0.)
C*   H1(I)=(1.,0.)
C*
C*   10
C*

```

ASDF6193  
ASDF6194  
ASDF6195  
ASDF6196  
ASDF6197  
ASDF6198  
ASDF6199  
ASDF6200  
ASDF6201  
ASDF6202  
ASDF6203  
ASDF6204  
ASDF6205  
ASDF6206  
ASDF6207  
ASDF6208  
ASDF6209  
ASDF6210  
ASDF6211  
ASDF6212  
ASDF6213  
ASDF6214  
ASDF6215  
ASDF6216  
ASDF6217  
ASDF6218  
ASDF6219  
ASDF6220  
ASDF6221  
ASDF6222  
ASDF6223  
ASDF6224  
ASDF6225  
ASDF6226  
ASDF6227  
ASDF6228  
ASDF6229  
ASDF6230  
ASDF6231  
ASDF6232  
ASDF6233  
ASDF6234  
ASDF6235  
ASDF6236  
ASDF6237  
ASDF6238  
ASDF6239  
ASDF6240





```

C* THE ROOTS COME FROM THE FORMER SUBROUTINE
C* THESE ARE ROOTS OF THE POLYNOMIALS N(Z), AND D(Z)
C* WHERE H(Z)=N(Z)/D(Z).
C* THE FACTORS OF THE POLYNOMIAL ARE OF THE FORM Z-ROOT
C*
      DO 15 K=1,11
      V(K)=-V(K)
15
C* WHEN K=N, WE HAVE GENERATED THE H1 COEFFICIENTS
C* OF N(Z), OR D(Z) OF H(Z)=N(Z)/D(Z)
C* THE REMAINING ITERATIONS ARE PERFORMED TO SIMPLIFY
C* OBTAINING COEFFICIENTS FOR N(Z**(-1)), AND D(Z**(-1)) LATER
C*
      DO 30 K=1,10
      "H2" REPRESENTS THE PREVIOUS RESULT TIMES Z**(-1)
C*
      DO 40 I=1,10
      H2(I+1)=H1(I)
      H2(1)=(0.,0.)
      DO 50 I=1,11
      H3(I)=H1(I)*V(I)
50
C* GENERATE THE NEW FINAL PRODUCT WITH K ROOTS MULTIPLIED.
C*
      DO 60 I=1,11
      H1(I)=H2(I)+H3(I)
60
C* GET THE NEXT ROOT TO BE MULTIPLIED AND PUT IT INTO "V(1)"
C*
      DO 70 I=1,10
      V(I)=V(I+1)
      CONTINUE
70
      DO 80 I=1,11
      H2(I)=H1(I)-I
      DO 90 I=1,11
      H1(I)=H2(I)
      RETURN
      END
80
90
95
      SUBROUTINE ADJUST(NNUM,NDEN)
      COMMON /ABVCTR/A(11),B(11)
      IDELTA=NDEN-NNUM
      ISTOP=11-IDELTA

```

ASD F6241  
 ASD F6242  
 ASD F6243  
 ASD F6244  
 ASD F6245  
 ASD F6246  
 ASD F6247  
 ASD F6248  
 ASD F6249  
 ASD F6250  
 ASD F6251  
 ASD F6252  
 ASD F6253  
 ASD F6254  
 ASD F6255  
 ASD F6256  
 ASD F6257  
 ASD F6258  
 ASD F6259  
 ASD F6260  
 ASD F6261  
 ASD F6262  
 ASD F6263  
 ASD F6264  
 ASD F6265  
 ASD F6266  
 ASD F6267  
 ASD F6268  
 ASD F6269  
 ASD F6270  
 ASD F6271  
 ASD F6272  
 ASD F6273  
 ASD F6274  
 ASD F6275  
 ASD F6276  
 ASD F6277  
 ASD F6278  
 ASD F6279  
 ASD F6280  
 ASD F6281  
 ASD F6282  
 ASD F6283  
 ASD F6284  
 ASD F6285  
 ASD F6286  
 ASD F6287  
 ASD F6288





ASDF6289  
ASDF6290  
ASDF6291  
ASDF6292  
ASDF6293  
ASDF6294  
ASDF6295  
ASDF6296  
ASDF6297  
ASDF6298  
ASDF6299  
ASDF6300  
ASDF6301  
ASDF6302  
ASDF6303  
ASDF6304  
ASDF6305  
ASDF6306  
ASDF6307  
ASDF6308  
ASDF6309  
ASDF6310  
ASDF6311  
ASDF6312  
ASDF6313  
ASDF6314  
ASDF6315  
ASDF6316  
ASDF6317  
ASDF6318  
ASDF6319  
ASDF6320  
ASDF6321  
ASDF6322  
ASDF6323  
ASDF6324  
ASDF6325  
ASDF6326  
ASDF6327  
ASDF6328  
ASDF6329  
ASDF6330  
ASDF6331  
ASDF6332  
ASDF6333  
ASDF6334  
ASDF6335  
ASDF6336

```

10  DO 10 I=1,ISTOP
    B(12-I)=B(12-I-IDELTA)
20  DO 20 K=1,IDELTA
    B(K)=0.
    RETURN
    END

    SUBROUTINE PLTIMP(ICHAR)

C* THIS SUBROUTINE PLOTS THE UNIT SAMPLE RESPONSE
C*

    DIMENSION PICT(1025)
    COMMON /IMPULS/ YI(1025), YS(1025), EN(1025)
    DO 5 K=1,1024
    EN(K+1)=K-1
    EN(1)=YI(1025)
    NFIN=YI(1025)
    DO 13 K=1,NFIN
    PICT(K+1)=YI(K)
    CONTINUE
    PICT(1)=YI(1025)
    CALL INIT
    CALL XFRM(2)
    CALL YFRM(2)
    IF(NFIN.LT.70)CALL LINE(-1)
    IF(NFIN.GE.70)CALL LINE(0)
    IF(NFIN.LT.70)CALL SYMBL(1)
    IF(NFIN.GE.70)CALL SYMBL(0)
    CALL SIZE(.4)
    CALL PLOT(EN,PICT)
    JFIN=YI(1025)+1
    IF(NFIN.GE.70)GO TO 40
    DO 30 K=2,JFIN
    CALL MOVEA(EN(K),PICT(K))
    CALL DRAWA(EN(K),0.0)
    CONTINUE
    CALL MOVEA(0.0,0.0)
    CALL DRAWA(EN(JFIN),0.0)
    CALL LABEL
    CALL VCURSR(ICHAR,X,Y)
    CALL FIN
    RETURN
    END

    SUBROUTINE PLTSTP(ICHAR)

C* THIS SUBROUTINE PLOTS THE UNIT STEP RESPONSE
C*

```



```

DIMENSION PICT(1025)
COMMON /IMPULS/ YI(1025), YS(1025), EN(1025)

```

5

```

XLAST=-500.
DO 5 K=1,1024
  EN(K+1)=K-1
  EN(1)=YS(1025)
  NFIN=YS(1025)
  DO 13 K=1,NFIN
    PICT(K+1)=YS(K)
  CONTINUE

```

13

```

  PICT(1)=NFIN
  CALL INIT
  CALL XFRM(2)
  CALL YFRM(2)
  IF(NFIN.LT.70)CALL LINE(-1)
  IF(NFIN.GE.70)CALL LINE(0)
  IF(NFIN.LT.70)CALL SYMBL(1)
  IF(NFIN.GE.70)CALL SYMBL(0)
  CALL SIZES(.4)
  CALL PLOT(EN,PICT)
  JFIN=NFIN+1 GO TO 40
  IF(NFIN.GE.70)GO TO 40
  DO 30 K=2,JFIN
    CALL MOVEA(EN(K),PICT(K))
    CALL DRAWA(EN(K),0.)
  CONTINUE
  CALL MOVEA(0.0,0.0)
  CALL DRAWA(EN(JFIN),0.0)
  CALL LABEL
  CALL VCURSR(ICCHAR,X,Y)
  CALL FIN
  RETURN
END

```

30  
40

```

SUBROUTINE PLITRF(IFNCTN,ISCALE,ICCHAR)

```

C\* THIS SUBROUTINE PLOTS BOTH THE PHASE AND  
C\* MAGNITUDE OF THE TRANSFER FUNCTION (LINEAR AND LOG)  
C\*

```

COMMON /FREQ/ YP(1026),YM(1026), PI(1026)
DO 10 K=1,1024
  PI(K+2)=FLOAT(K-1)*3.14/1024.
  PI(1)=1025.
  PI(2)=0.

```

10

```

  CALL INIT
  CALL XFRM(2)
  CALL YFRM(2)
  IF(IFNCTN.EQ.3)CALL PLOT(PI,YP)

```

ASDF6337  
ASDF6338  
ASDF6339  
ASDF6340  
ASDF6341  
ASDF6342  
ASDF6343  
ASDF6344  
ASDF6345  
ASDF6346  
ASDF6347  
ASDF6348  
ASDF6349  
ASDF6350  
ASDF6351  
ASDF6352  
ASDF6353  
ASDF6354  
ASDF6355  
ASDF6356  
ASDF6357  
ASDF6358  
ASDF6359  
ASDF6360  
ASDF6361  
ASDF6362  
ASDF6363  
ASDF6364  
ASDF6365  
ASDF6366  
ASDF6367  
ASDF6368  
ASDF6369  
ASDF6370  
ASDF6371  
ASDF6372  
ASDF6373  
ASDF6374  
ASDF6375  
ASDF6376  
ASDF6377  
ASDF6378  
ASDF6379  
ASDF6380  
ASDF6381  
ASDF6382  
ASDF6383  
ASDF6384



```

15 IF(ISCAL.EQ.1)GO TO 20
DO 15 I=3,1026
YM(I)=20.*ALOG10(YM(I))
20 YM(2)=.01
IF(IFNCIN.EQ.4)CALL PLOT(PI,YM)
CALL LABEL
CALL TSEND
CALL VCURSR(ICHAR,X,Y)
CALL FIN
RETURN
END

```

```

C* SUBROUTINE RSPNSE (IFNCIN,FLTRGN)
C* THIS SUBROUTINE COMPUTES THE UNIT SAMPLE RESPONSE OF THE SYSTEM
C*
C* THE "Z" VECTOR HOLDS THE DELAYED VALUES
C* FOR THE RECURSIVE CALCULATION.
C*

```

```

      DIMENSION Z(11)
      REAL INPUT

```

```

C* "ABVCTR" IS THE PAIR OF VECTORS HOLDING
C* THE VALUES OF A(K), AND B(K)
C*

```

```

      COMMON /ABVCTR/ A(11), B(11)

```

```

C* "IMPULS" IS THE LABELED COMMON HOLDING THE UNIT SAMPLE RESPONSE
C*
C* COMMON /IMPULS/ YI(1025), YS(1025), EN(1025)

```

```

C* INITIALIZE THE Z VECTOR
C*

```

```

      XLAST=0.
      XMIN=0.
      XMAX=0.
      DELTA=0.
      DO 5 K=1,11
      Z(K)=0.
5

```

```

C* COMPUTE THE 1024 POINT UNIT SAMPLE OR STEP RESPONSE
C*

```

```

      DO 30 N=1,1024

```

```

C* EACH VALUE OF XSUBN WILL BE ONE EXCEPT FOR THE CASE
C* OF THE UNIT SAMPLE RESPONSE WHEN N>1, AND THEN XSUBN=0.
C*
      XSUBN = 1.

```

ASDF6385  
ASDF6386  
ASDF6387  
ASDF6388  
ASDF6389  
ASDF6390  
ASDF6391  
ASDF6392  
ASDF6393  
ASDF6394  
ASDF6395  
ASDF6396  
ASDF6397  
ASDF6398  
ASDF6399  
ASDF6400  
ASDF6401  
ASDF6402  
ASDF6403  
ASDF6404  
ASDF6405  
ASDF6406  
ASDF6407  
ASDF6408  
ASDF6409  
ASDF6410  
ASDF6411  
ASDF6412  
ASDF6413  
ASDF6414  
ASDF6415  
ASDF6416  
ASDF6417  
ASDF6418  
ASDF6419  
ASDF6420  
ASDF6421  
ASDF6422  
ASDF6423  
ASDF6424  
ASDF6425  
ASDF6426  
ASDF6427  
ASDF6428  
ASDF6429  
ASDF6430  
ASDF6431  
ASDF6432





ASDF6433  
ASDF6434  
ASDF6435  
ASDF6436  
ASDF6437  
ASDF6438  
ASDF6439  
ASDF6440  
ASDF6441  
ASDF6442  
ASDF6443  
ASDF6444  
ASDF6445  
ASDF6446  
ASDF6447  
ASDF6448  
ASDF6449  
ASDF6450  
ASDF6451  
ASDF6452  
ASDF6453  
ASDF6454  
ASDF6455  
ASDF6456  
ASDF6457  
ASDF6458  
ASDF6459  
ASDF6460  
ASDF6461  
ASDF6462  
ASDF6463  
ASDF6464  
ASDF6465  
ASDF6466  
ASDF6467  
ASDF6468  
ASDF6469  
ASDF6470  
ASDF6471  
ASDF6472  
ASDF6473  
ASDF6474  
ASDF6475  
ASDF6476  
ASDF6477  
ASDF6478  
ASDF6479  
ASDF6480

```

IF(N.GT.1.AND.(FNCTN.EQ.0)XSUBN = 0.
INPUT=XSUBN
DO 10 K=2,11
INPUT=INPUT-A(K)*Z(K)
Z(1)=INPUT
OUTPUT=0.
DO 20 K=1,11
OUTPUT=OUTPUT+B(K)*Z(K)
DO 25 K=1,10
Z(12-K)=Z(11-K)
IEND=N
C* TEST TO SEE IF A STEADY STATE HAS BEEN ACHIEVED.
C*
C* 27 IF(ABS(OUTPUT).GT.XMAX)XMAX=ABS(OUTPUT)
DIFFER=.01*XMAX
IF(ABS(OUTPUT-XLAST).GT.DIFFER)DELTA=0.
XLAST=OUTPUT
DELTA=DELTA+1
IF(DELTA.GT.20)GO TO 38
IF(FNCTN.EQ.0)YI(N)=OUTPUT*FLTRGN
IF(FNCTN.EQ.1)YS(N)=OUTPUT*FLTRGN
30 IF(FNCTN.EQ.0)YI(1025)=IEND-15
38 IF(FNCTN.EQ.1)YS(1025)=IEND-15
999 RETURN
END

SUBROUTINE FRQNCY(NNUM,NDEN,FLTRGN)
C* FREQUENCY COMPUTES THE FREQUENCY RESPONSE FOR
C* THE FILTER BEING ANALYZED.
C*
COMMON /ABVCTR/A(11), B(11)
COMMON /FREQ/YP(1026),YM(1026)
REAL*8 X,Y,THETA,ZRO,ONE,AA(11),BB(11),XX,G(2)
COMPLEX*16 Z,FRCTOP,FRCBOT,HOFZ,ZEXP,TERM,TERMD
EQUIVALENCE (G,HOFZ)
ZRO=0.0D+00
ONE=1.0D+00

C* MAKE THE COEFFICIENTS DOUBLE PRECISION
C*
C* DO 20 K=1,11
AA(K)=A(K)
BB(K)=B(K)
CONTINUE
20
C* COMPUTE PHASE AND MAGNITUDE OF H(Z)

```





```

C*      DO 60 K=1,1025
        THETA=FLOAT(K-1)*3.14159265/1024.
        Z=CDEXP(DCMPLX(ZRO,THETA))
25      FRCBOT=0.
C*      COMPUTE THE VALUE OF THE NUMERATOR OF H(Z)
C*
C*      DO 40 N=1,NNUM
        ZEXP=Z**(N-1)
        TERM=DCMPLX(BB(N),ZRO)/ZEXP
        FRCBOT=FRCBOT+TERM
40      CONTINUE
C*      COMPUTE THE VALUE OF THE DENOMINATOR FACTORS OF H(Z)
C*
C*      DO 50 N=2,NDEN
        TERM=DCMPLX(AA(N),ZRO)/Z**(N-1)
        FRCBOT=FRCBOT+TERM
50      CONTINUE
C*      GET H(Z) IN FINAL FORM
C*
C*      HOFZ=FRCBOT/(DCMPLX(ONE,ZRO)+FRCBOT)
C*      COMPUTE THE MAGNITUDE OF H(Z)
C*
C*      YM(K+2)=CDABS(HOFZ)
        YM(K+2)=YM(K+2)*FLTRGN
55      C*      COMPUTE THE PHASE OF H(Z)
C*
C*      X=G(1)
        Y=G(2)
        IF(X.EQ.ZRO.AND.Y.EQ.ZRO)YP(K+2)=YP(K+1)
        IF(X.EQ.ZRO.AND.Y.EQ.ZRO)GO TO 60
        XX=DATAN2(Y,X)
        YP(K+2)=XX
        CONTINUE
        YP(1)=1025.
        YP(2)=0.
        YM(1)=1025.
        YM(2)=0.
        RETURN
        END
60

```



SUBROUTINE FLIR2(FLTRGN)

C\*  
C\*  
C\*  
C\*

FLIR2 WRITES A LISTING OF ALL  
POLE AND ZERO LOCATIONS FOR REVIEW BEFORE  
PROCEEDING WITH THE PLOTTING OF RESPONSES

COMMON IROOTS(4), POLZRO(20,5)

DIMENSION IRTTP(5)

DATA IZ/1HZ/, IP/1HP/, IZP/2HZP/

DATA IRTTP/2H , 2HRZ, 2HCL, 2HRP, 2HCP/

CALL INIT

CALL FIN

WRITE(6,102) (IZ, I=1,70)

FORMAT (1X,70A1)

WRITE (6,108)

108 FORMAT (1X,Z, 5X, 'REAL', 6X, 'Z', 3X, 'IMAGINARY', 3X, 'Z',  
1, 6X, 'RHO', 6X, 'Z', 5X, 'THEIA', 5X, 'Z RT Z',)

WRITE (6,102) (IZ, I= 1,70)

DO 10 I=1,10

ITYPEP=POLZRO(I,5)+1.

IF (POLZRO(I,5).NE.2.) AND. POLZRO(I,5).NE.4.)

1WRITE(6,104) (POLZRO(I,J), J=1,4), IRTTP(ITYPEP)

IF (POLZRO(I,5).EQ.2.) OR. POLZRO(I,5).EQ.4.)

1WRITE(6,103) (POLZRO(I,J), J=1,4), IRTTP(ITYPEP)

FORMAT(1X,Z, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

13X, Z, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

FORMAT(1X,Z, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

1, F10.7, F10.7, Z, F10.7, F10.7, Z, F10.7,

ASDF6529  
ASDF6530  
ASDF6531  
ASDF6532  
ASDF6533  
ASDF6534  
ASDF6535  
ASDF6536  
ASDF6537  
ASDF6538  
ASDF6539  
ASDF6540  
ASDF6541  
ASDF6542  
ASDF6543  
ASDF6544  
ASDF6545  
ASDF6546  
ASDF6547  
ASDF6548  
ASDF6549  
ASDF6550  
ASDF6551  
ASDF6552  
ASDF6553  
ASDF6554  
ASDF6555  
ASDF6556  
ASDF6557  
ASDF6558  
ASDF6559  
ASDF6560  
ASDF6561  
ASDF6562  
ASDF6563  
ASDF6564  
ASDF6565  
ASDF6566  
ASDF6567  
ASDF6568  
ASDF6569  
ASDF6570  
ASDF6571  
ASDF6572  
ASDF6573  
ASDF6574  
ASDF6575  
ASDF6576



```

111 FORMAT(20X,'NUMBER OF COMPLEX ZERO PAIRS:',I3)
    WRITE(6,113) IROOTS(3)
113 FORMAT(20X,'NUMBER OF REAL POLES:',I3)
    WRITE(6,114) IROOTS(4)
114 FORMAT(20X,'NUMBER OF COMPLEX POLE PAIRS:',I3)
    WRITE(6,117) FLTRGN
117 FORMAT(25X,'FILTER GAIN:',F12.8)
    WRITE(6,115)
115 FORMAT(' HIT SPACE THEN RETURN')
    READ(5,116)
116 FORMAT(A1)
    RETURN
    END

```

```

SUBROUTINE LABLE
C* SUBROUTINE LABLE WRITES THE LABLES ON EACH
C* OF THE FIVE PLOTS DRAWN
C*

```

```

    DIMENSION ITITLE(40)
    CALL ANMODE
    READ(2,100) IX,IY,INUM,(ITITLE(J),J=1,40)
    CALL RECOVR
    CALL MOVABS(IX,IY)
    CALL ANMODE
    WRITE(6,101) (ITITLE(J),J=1,INUM)
101 FORMAT(1X,30A1)
    READ(2,100) IX,IY,INUM,(ITITLE(J),J=1,40)
    DO 10 I=1,INUM
    CALL RECOVR
    CALL MOVABS(IX,IY)
    CALL ANMODE

```

```

    WRITE(6,102) ITITLE(I)
102 FORMAT(1X,A1)
    IY=IY-22
    READ(2,100) IX,IY,INUM,(ITITLE(J),J=1,40)
    CALL RECOVR
    CALL MOVABS(IX,IY)
    CALL ANMODE
    WRITE(6,101) (ITITLE(J),J=1,INUM)
    READ(2,100) IX,IY,INUM,(ITITLE(J),J=1,40)
    CALL RECOVR
    CALL MOVABS(IX,IY)
    CALL ANMODE
    WRITE(6,101) (ITITLE(J),J=1,INUM)
    CALL RECOVR
    RETURN
    END

```

ASDF6577  
ASDF6578  
ASDF6579  
ASDF6580  
ASDF6581  
ASDF6582  
ASDF6583  
ASDF6584  
ASDF6585  
ASDF6586  
ASDF6587  
ASDF6588  
ASDF6589  
ASDF6590  
ASDF6591  
ASDF6593  
ASDF6594  
ASDF6595  
ASDF6596  
ASDF6597  
ASDF6598  
ASDF6599  
ASDF6600  
ASDF6601  
ASDF6602  
ASDF6603  
ASDF6604  
ASDF6605  
ASDF6606  
ASDF6607  
ASDF6608  
ASDF6609  
ASDF6610  
ASDF6611  
ASDF6612  
ASDF6613  
ASDF6614  
ASDF6615  
ASDF6616  
ASDF6617  
ASDF6618  
ASDF6619  
ASDF6620  
ASDF6621  
ASDF6622  
ASDF6623  
ASDF6624  
ASDF6625





# APPENDIX I: SOURCE DECK FOR ASDF COMMAND: HRD\$CPY

```

C***** SOURCE DECK FOR ASDF COMMAND: HRD$CPY *****
C*
C*
C***** COMMON /ABVCTR/ A(11), B(11), EN(1026), YM(1026), YMDB(1026) *****
C***** COMMON /IMPULS/ YI(1026), YS(1026), PI(1026), YMDB(1026) *****
C***** COMMON /FREQ/ YP(1026), YM(1026), PI(1026), YMDB(1026) *****
C***** CALL ERRSET(208,256,-1,1,1) *****
C***** LINEAR=1 *****
C***** LOG=2 *****
C*
C* INITIALIZE A AND B TO ZERO
C*
C*
C* DO 5 K=1,20
C* A(K)=0.
C* B(K)=0.
C*
C* READ THE POLZRO TABLE OFF OF THE FILE
C*
C* DO 20 I=1,20
C* READ(5,100) (POLZRO(I,J),J=1,5)
C* 100 FORMAT(1X,5F12.8)
C*
C* READ THE IROOTS VALUES, AND FILTER GAIN FROM THE FILE
C*
C* READ(5,101) (IROOTS(I),I=1,4)
C* 101 FORMAT(1X,4I2)
C* READ(5,102) FLTRGN
C* 102 FORMAT(1X,F12.8)
C* FLTRGN=ABS(FLTRGN)
C*
C* PRINT OUT A TABLE OF THE POLES AND ZEROS
C*
C* CALL FLTR2(FLTRGN)
C*
C* GENERATE THE A(K) COEFFICIENTS
C*
C* CALL ASUBK(A,NORDER)
C* NDN=NORDER+1
C*
C* GENERATE THE B(R) COEFFICIENTS
C*
C* CALL BSUBR(B,VORDER)
C* NNUM=NORDER+1
C*
C* CHECK TO SEE IF THE FILTER IS CAUSAL

```





```

C*      IF(NNUM.GT.NDEN)WRITE(6,103)
103     FORMAT(10('/'),' FILTER IS NOT CAUSAL - PROGRAM TERMINATES')
C*      IF(NNUM.GT.NDEN)STOP
C*      ADJUST THE COEFFICIENTS IF NNUM DOES NOT EQUAL NDEN
C*      IF(NNUM.NE.NDEN)CALL ADJUST(NNUM,NDEN)
C*      COMPUTE THE UNIT SAMPLE RESPONSE
C*      CALL RSPNSE(0,FLTRGN)
C*      CALCULATE THE STEP RESPONSE
C*      CALL RSPNSE(1,FLTRGN)
C*      CALCULATE THE FREQUENCY RESPONSE
C*      CALL FRQNCY(NNUM,NDEN,FLTRGN)
C*      CALL PRPPLT(IYIPTS,IYSPTS,NUMPLT)
C*      PLOT THE UNIT SAMPLE RESPONSE
C*      CALL PLTSTP(IYIPTS,FLTRGN,NUMPLT)
C*      PLOT THE STEP RESPONSE
C*      CALL PLTSTP(IYSPTS,FLTRGN,NUMPLT)
C*      PLOT THE PHASE OF THE TRANSFER FUNCTION
C*      CALL PLTTRF(3,LINEAR,FLTRGN,NUMPLT)
C*      PLOT THE MAGNITUDE OF THE TRANSFER FUNCTION
C*      CALL PLTTRF(4,LINEAR,FLTRGN,NUMPLT)
C*      CALL PLTTRF(4,LOG,FLTRGN,NUMPLT)
C*      PLOT THE UNIT CIRCLE WITH POLES AND ZEROS
C*      CALL UNTCR(NUMPLT)
C*      OUTPUT THE DATA
C*      CALL OUTNUM
C*      STOP
C*      END

```

```

ASDF7049
ASDF7050
ASDF7051
ASDF7052
ASDF7053
ASDF7054
ASDF7055
ASDF7056
ASDF7057
ASDF7058
ASDF7059
ASDF7060
ASDF7061
ASDF7062
ASDF7063
ASDF7064
ASDF7065
ASDF7066
ASDF7067
ASDF7068
ASDF7069
ASDF7070
ASDF7071
ASDF7072
ASDF7073
ASDF7074
ASDF7075
ASDF7076
ASDF7077
ASDF7078
ASDF7079
ASDF7080
ASDF7081
ASDF7082
ASDF7083
ASDF7084
ASDF7085
ASDF7086
ASDF7087
ASDF7088
ASDF7089
ASDF7090
ASDF7091
ASDF7092
ASDF7093
ASDF7094
ASDF7095
ASDF7096

```



ASDF7097  
ASDF7098  
ASDF7099  
ASDF7100  
ASDF7101  
ASDF7102  
ASDF7103  
ASDF7104  
ASDF7105  
ASDF7106  
ASDF7107  
ASDF7108  
ASDF7109  
ASDF7110  
ASDF7111  
ASDF7112  
ASDF7113  
ASDF7114  
ASDF7115  
ASDF7116  
ASDF7117  
ASDF7118  
ASDF7119  
ASDF7120  
ASDF7121  
ASDF7122  
ASDF7123  
ASDF7124  
ASDF7125  
ASDF7126  
ASDF7127  
ASDF7128  
ASDF7129  
ASDF7130  
ASDF7131  
ASDF7132  
ASDF7133  
ASDF7134  
ASDF7135  
ASDF7136  
ASDF7137  
ASDF7138  
ASDF7139  
ASDF7140  
ASDF7141  
ASDF7142  
ASDF7143  
ASDF7144

```

SUBROUTINE ADJUST(NNUM,NDEN)
COMMON /ABVCTR/A(11),B(11)
IDELTA=NDEN-NNUM
ISTOP=11-IDELTA
DO 10 I=1,ISTOP
  B(12-I)=B(12-I-IDELTA)
DO 20 K=1,IDELTA
  B(K)=0.
NNUM=NDEN
RETURN
END

```

10

20

SUBROUTINE OUTNUM

C\* OUTNUM PRINTS OUT THE NUMERICAL VALUES  
C\* OF THE FILTER'S RESPONSES  
C\*

```

COMMON /IMPULS/ YI(1026), YS(1026), EN(1026)
COMMON /FREQ/ YP(1026), YM(1026), PI(1026), YMDB(1026)
DIMENSION M(1026)
DO 20 L=1,1026
  M(L)=L-1
DO 40 I=1,18
  CALL HEADNG
DO 40 J=1,6
  IBEG=(I-1)*60+(J-1)*10+1
  IF(I.EQ.18.AND.J.EQ.1)IEND=1024
  WRITE(6,104)(M(K),YI(K),YS(K),YP(K),YM(K),PI(K),K=IBEG,IEND)
  FORMAT(10(2X,I6,2E20.8,3E21.8,/,/,))
  IF(I.EQ.18.AND.J.EQ.1)GO TO 41
  CONTINUE
  WRITE(6,105)
  FORMAT(1H1)
  RETURN
END

```

20

104

40

41

105

SUBROUTINE HEADNG

C\* HEADNG PRINTS THE PAGE HEADINGS  
C\*

```

DATA IX/1HX/
WRITE(6,100)
FORMAT(1H1)
WRITE(6,101)(IX,I=1,114)
FORMAT(1X,114A1)
WRITE(6,108)

```

100

101









```

HDLTA=.2
Y=1.5
DO 80 I=1,23
X=5.4
IF(I.NE.7.AND.I.NE.17.AND.I.NE.2.AND.I.NE.22)GO TO 81
X=5.3
CALL PLOT(X,Y,IUP)
X=5.7
CALL PLOT(X,Y,IDOWN)
GO TO 80
CALL PLOT(X,Y,IUP)
X=X+HDLTA
CALL PLOT(X,Y,IDOWN)
Y=Y+VDLTA
CALL PLOT(UCX(1),UCY(1),+3)
DO 60 I=1,360
CALL PLOT(UCX(1),UCY(1),+2)
CALL PLOT(UCX(1),UCY(1),+2)
CALL SYMBOL(-1.,+2.8125,-1875,11HUNIT CIRCLE,0.,11)
CALL SYMBOL(-2.4,-3.,1875,26HFILTER POLE ZERO LOCATIONS,0.,26)
DO 30 I=1,20
DO 30 J=1,3
POLZRO(I,J)=POLZRO(I,J)*2.5
DO 40 I=1,10
IF(POLZRO(I,3).EQ.-25.)GO TO 40
IF(POLZRO(I,3).LE.2.5)GO TO 45
POLZRO(I,1)=2.625*COS(POLZRO(I,4))
POLZRO(I,2)=2.625*SIN(POLZRO(I,4))
CALL SYMBOL(POLZRO(I,1),POLZRO(I,2),.25,11,0.,-1)
IF(POLZRO(I,5).NE.1.)CALL SYMBOL(POLZRO(I,1),POLZRO(I,2),.25,
X11,0.,-1)
GO TO 40
CALL SYMBOL(POLZRO(I,1),POLZRO(I,2),.25,1,0.,-1)
IF(POLZRO(I,5).NE.1.)CALL SYMBOL(POLZRO(I,1),POLZRO(I,2),.25,
X1,0.,-1)
CONTINUE
DO 50 I=1,20
IF(POLZRO(I,1).EQ.-25.)GO TO 50
CALL SYMBOL(POLZRO(I,1),POLZRO(I,2),.25,4,0.,-1)
IF(POLZRO(I,5).NE.3)CALL SYMBOL(POLZRO(I,1),POLZRO(I,2),.25,4,0.
X,-1)
CONTINUE
CALL PLOT(0.,0.,+999)
RETURN
END
SUBROUTINE ASJBK (POLE,NORDER)

```





```

C* GENERATE THE AK COEFFICIENTS FROM THE POLES
C*
C* COMMON IROOTS(4), POLZRO(20,5), IERROR
C* COMPLEX POLES(11), ACDEF(11)
C* REAL POLE(11)
C*
C* INITIALIZE ALL THE VECTORS AND INDICES TO ZERO
C*
C*
C* DO 5 K=1,11
C* POLES(K)=(0.,0.)
C* POLE(K)=0.
C* ACDEF(K)=0.
C*
C* "J" INDEXES THE TABLE "POLES"
C* "K" INDEXES THE TABLE "POLZRO"
C*
C* J=1
C* K=1
C*
C* COMPUTE THE ORDER OF THE SYSTEM OF POLES
C*
C* NORDER=IROOTS(3)+2*IROOTS(4)
C* IF(NORDER.NE.0)GO TO 10
C* POLE(1)=1.0
C* RETURN
C*
C* THE VECTOR "POLES" WILL CONTAIN A LIST OF ALL
C* THE POLES, IE., X+J0, X+JY, X-JY, . . .
C*
C* 10 POLES(J)=CMPLX(POLZRO(10+K,1),POLZRO(10+K,2))
C*
C* TEST TO SEE IF ALL ROOTS HAVE BEEN
C* IF(J.EQ.NORDER)GO TO 20
C* J=J+1
C*
C* IF THE ROOT IS REAL CONTINUE PROCESING THE NEXT ROOT
C*
C* IF(POLZRO(K+10,5).NE.3)GO TO 15
C* K=K+1
C* GO TO 10
C*
C* SINCE THE ROOT IS COMPLEX PUT THE CONJUGATE
C* OF THE ROOT IN THE "POLES" TABLE.
C*
C* 15 POLES(J)=CMPLX(POLZRO(10+K,1),-POLZRO(10+K,2))
C* IF(J.EQ.NORDER) GO TO 20

```



```

J=J+1
K=K+1
GO TO 10

C* GENERATE THE COEFFICIENTS FOR THE D(Z) POLYNOMIAL
C* OF THE TRANSFER FUNCTION H(Z).
C* 20 CALL COEFF(POLES,NORDER,ACOE)
C* CHANGE THE COEFFICIENTS FROM COMPLEX ASUBK TO REAL ASUBK
C* 40 DO 40 J=1,11
C* POLE(J)=REAL(ACOE(J))
C* RETURN
C* END

C* SUBROUTINE BSUBR(ZERO,NORDER)
C* GENERATE THE B(R) COEFFICIENTS FROM THE ZEROS
C* COMMON IROOTS(4), POLZRO(20,5), IERROR
C* COMPLEX ZEROS(11), BCOEF(11)
C* REAL ZERO(11)

C* INITIALIZE ALL VECTORS AND INDICES
C* 5 DO 5 K=1,11
C* ZEROS(K)=(0.,0.)
C* ZERO(K)=0.
C* BCOEF(K)=(0.,0.)
C* "J" INDEXES THE "ZEROS" TABLE
C* "K" INDEXES THE "POLZRO" TABLE
C* J=1
C* K=1

C* COMPUTE THE ORDER OF THE SYSTEM OF ZEROS.
C* NORDER=IROOTS(1)+2*IROOTS(2)
C* IF(NORDER.NE.0) GO TO 10
C* ZERO(1)=1.0
C* RETURN

C* THE VECTOR "ZEROS" CONTAINS A LIST OF ALL
C* OF THE ZEROS, IE., X+J0, X+JY, X-JY, . . .
C* 10 ZEROS(J)=CMPLX(POLZRO(K,1),POLZRO(K,2))

```

```

ASDF7290
ASDF7291
ASDF7292
ASDF7293
ASDF7294
ASDF7295
ASDF7296
ASDF7297
ASDF7298
ASDF7299
ASDF7300
ASDF7301
ASDF7302
ASDF7303
ASDF7304
ASDF7305
ASDF7306
ASDF7307
ASDF7308
ASDF7309
ASDF7310
ASDF7311
ASDF7312
ASDF7313
ASDF7314
ASDF7315
ASDF7316
ASDF7317
ASDF7318
ASDF7319
ASDF7320
ASDF7321
ASDF7322
ASDF7323
ASDF7324
ASDF7325
ASDF7326
ASDF7327
ASDF7328
ASDF7329
ASDF7330
ASDF7331
ASDF7332
ASDF7333
ASDF7334
ASDF7335
ASDF7336
ASDF7337

```



```

C* CHECK TO SEE IF ALL THE ROOTS HAVE BEEN PROCESSED
C*
C* IF(J.EQ.NORDER)GO TO 20
C* J=J+1
C*
C* IF THE ROOT IS REAL CONTINUE PROCESSING THE NEXT ROOT
C*
C* IF(POLZRO(K,5).NE.1)GO TO 15
C* K=K+1
C* GO TO 10
C*
C* SINCE THE ROOT IS COMPLEX, ENTER THE CONJUGATE
C* INTO THE TABLE "ZEROS".
C*
C* 15 ZEROS(J)=CMPLX(POLZRO(K,1),-POLZRO(K,2))
C* IF(J.EQ.NORDER)GO TO 20
C* J=J+1
C* K=K+1
C* GO TO 10
C*
C* GENERATE THE BSUBR COEFFICIENTS
C*
C* 20 CALL COEFF(ZEROS, NORDER, BCOEF)
C*
C* MAKE THE COEFFICIENTS BSUBR REAL.
C*
C* 40 DO 40 J=1,11
C* ZERO(J)=REAL(BCOEF(J))
C* RETURN
C* END
C*
C* SUBROUTINE COEFF(V,N,H1)
C*
C* "V" - VECTOR OF ROOTS; "N" - THE ORDER OF THE SYSTEM
C* "H1" - VECTOR WITH THE FINAL COEFFICIENTS
C* H(1)=Z***(0), H1(2)=Z***(-1), . . . ,H1(11)=Z***(-10)
C*
C* COMPLEX H1(11),H2(11),H3(11),V(11)
C* INITIALIZE VECTORS
C* IZERO=0
C* DO 10 I=1,11
C* H1(I)=(0.,0.)
C* H2(I)=(0.,0.)
C* H3(I)=(0.,0.)
C* H1(I)=(1.,0.)
C*
C* 10
C*
C* THE ROOTS COME FROM THE FORMER SUBROUTINE

```

ASDF7338  
ASDF7339  
ASDF7340  
ASDF7341  
ASDF7342  
ASDF7343  
ASDF7344  
ASDF7345  
ASDF7346  
ASDF7347  
ASDF7348  
ASDF7349  
ASDF7350  
ASDF7351  
ASDF7352  
ASDF7353  
ASDF7354  
ASDF7355  
ASDF7356  
ASDF7357  
ASDF7358  
ASDF7359  
ASDF7360  
ASDF7361  
ASDF7362  
ASDF7363  
ASDF7364  
ASDF7365  
ASDF7366  
ASDF7367  
ASDF7368  
ASDF7369  
ASDF7370  
ASDF7371  
ASDF7372  
ASDF7373  
ASDF7374  
ASDF7375  
ASDF7376  
ASDF7377  
ASDF7378  
ASDF7379  
ASDF7380  
ASDF7381  
ASDF7382  
ASDF7383  
ASDF7384  
ASDF7385





```

C* THESE ARE ROOTS OF THE POLYNOMIALS N(Z), AND D(Z)
C* WHERE  $H(Z) = N(Z)/D(Z)$ .
C* THE FACTORS OF THE POLYNOMIAL ARE OF THE FORM Z-ROOT
C*
C* 15 DO 15 K=1,11
C* V(K)=-V(K)
C*
C* WHEN K=N, WE HAVE GENERATED THE H1 COEFFICIENTS
C* OF N(Z), OR D(Z) OF  $H(Z) = N(Z)/D(Z)$ 
C* THE REMAINING ITERATIONS ARE PERFORMED TO SIMPLIFY
C* OBTAINING COEFFICIENTS FOR  $N(Z^{**}-1)$ , AND  $D(Z^{**}-1)$  LATER
C*
C* 30 DO 30 K=1,10
C*
C* "H2" REPRESENTS THE PREVIOUS RESULT TIMES  $Z^{**}(-1)$ 
C*
C* 40 DO 40 I=1,10
C* H2(I+1)=H1(I)
C* H2(1)=(0.,0.)
C* DO 50 I=1,11
C* H3(I)=H1(I)*V(I)
C*
C* GENERATE THE NEW FINAL PRODUCT WITH K ROOTS MULTIPLIED.
C*
C* 60 DO 60 I=1,11
C* H1(I)=H2(I)+H3(I)
C*
C* GET THE NEXT ROOT TO BE MULTIPLIED AND PUT IT INTO "V(1)"
C*
C* 65 DO 70 I=1,10
C* V(I)=V(I+1)
C* 70 CONTINUE
C*
C* TO GET THE COEFFICIENTS FOR  $H(Z^{**}-1) = N(Z^{**}-1)/D(Z^{**}-1)$ 
C* THE ORDER OF THE COEFFICIENTS MUST BE REVERSED.
C*
C* 80 DO 80 I=1,11
C* H2(I)=H1(12-I)
C* DO 90 I=1,11
C* H1(I)=H2(I)
C* 90 RETURN
C* 95 END
C*
C* SUBROUTINE RSPNSE (IFNCTN,FLTRGN)
C* THIS SUBROUTINE COMPUTES THE UNIT SAMPLE RESPONSE OF THE SYSTEM
C* THE "Z" VECTOR HOLDS THE DELAYED VALUES

```

ASDF7386  
ASDF7387  
ASDF7388  
ASDF7389  
ASDF7390  
ASDF7391  
ASDF7392  
ASDF7393  
ASDF7394  
ASDF7395  
ASDF7396  
ASDF7397  
ASDF7398  
ASDF7399  
ASDF7400  
ASDF7401  
ASDF7402  
ASDF7403  
ASDF7404  
ASDF7405  
ASDF7406  
ASDF7407  
ASDF7408  
ASDF7409  
ASDF7410  
ASDF7411  
ASDF7412  
ASDF7413  
ASDF7414  
ASDF7415  
ASDF7416  
ASDF7417  
ASDF7418  
ASDF7419  
ASDF7420  
ASDF7421  
ASDF7422  
ASDF7423  
ASDF7424  
ASDF7425  
ASDF7426  
ASDF7427  
ASDF7428  
ASDF7429  
ASDF7430  
ASDF7431  
ASDF7432  
ASDF7433





```

C* FOR THE RECURSIVE CALCULATION.
C*
  REAL*8 XMIN,XMAX,XLAST,XSUBN,INPUT,OUTPUT,AA(11),BB(11),
  XZ(11),DIFFER,DFTRGN,ZRO,ONE,PIDONE
C*
C* "ABVCTR" IS THE PAIR OF VECTORS HOLDING
C* THE VALUES OF A(K), AND B(K)
C*
  COMMON /ABVCTR/ A(11), B(11)
C*
C* "IMPULS" IS THE LABELED COMMON HOLDING THE UNIT SAMPLE RESPONSE
C*
  COMMON /IMPULS/ YI(1026), YS(1026), EN(1026)
C*
C* INITIALIZE THE CONSTANTS AND Z VECTOR
C*
  IFLAG=0
  ZRO=0.0D+00
  ONE=1.0D+00
  PIDONE=1.0D-02
  XMIN=ZRO
  XMAX=ZRO
  XLAST=ZRO
  DELTA=0.
  DFTRGN=FLTRGN
  DO 4 K=1,11
    AA(K)=A(K)
    BB(K)=B(K)
    DO 5 K=1,11
      Z(K)=0.
4
5
C* COMPUTE THE 1024 POINT UNIT SAMPLE OR STEP RESPONSE
C*
  DO 30 N=1,1024
C*
C* EACH VALUE OF XSUBN WILL BE ONE EXCEPT FOR THE CASE
C* OF THE UNIT SAMPLE RESPONSE WHEN N>1, AND THEN XSUBN=0.
C*
  XSUBN=ONE
  IF(N.GT.1.AND.(IFNCTN.EQ.0))XSUBN=ZRO
  INPUT=XSUBN
  DO 10 K=2,11
    INPUT=INPUT-A(K)*Z(K)
  Z(1)=INPUT
  OUTPUT=ZRO
  DO 20 K=1,11
    OUTPUT=OUTPUT+B(K)*Z(K)
  DO 25 K=1,10

```

ASDF7434  
 ASDF7435  
 ASDF7436  
 ASDF7437  
 ASDF7438  
 ASDF7439  
 ASDF7440  
 ASDF7441  
 ASDF7442  
 ASDF7443  
 ASDF7444  
 ASDF7445  
 ASDF7446  
 ASDF7447  
 ASDF7448  
 ASDF7449  
 ASDF7450  
 ASDF7451  
 ASDF7452  
 ASDF7453  
 ASDF7454  
 ASDF7455  
 ASDF7456  
 ASDF7457  
 ASDF7458  
 ASDF7459  
 ASDF7460  
 ASDF7461  
 ASDF7462  
 ASDF7463  
 ASDF7464  
 ASDF7465  
 ASDF7466  
 ASDF7467  
 ASDF7468  
 ASDF7469  
 ASDF7470  
 ASDF7471  
 ASDF7472  
 ASDF7473  
 ASDF7474  
 ASDF7475  
 ASDF7476  
 ASDF7477  
 ASDF7478  
 ASDF7479  
 ASDF7480  
 ASDF7481



```

25      Z(12-K)=Z(11-K)
C*      IEND=N
C*
C*      TEST TO SEE IF A STEADY STATE HAS BEEN ACHIEVED.
C*
      IF(DABS(OUTPUT).GT.XMAX)XMAX=DABS(OUTPUT)
      DIFFER=PTOONE*XMAX
      IF(DABS(OUTPUT-XLAST).GT.DIFFER)DELTA=0.
      XLAST=OUTPUT
      DELTA=DELTA+1
      IF(IFUNCTN.EQ.0)YI(N)=OUTPUT*DFTRGN
      IF(IFUNCTN.EQ.1)YS(N)=OUTPUT*DFTRGN
      IF(DELTA.LE.20.)GO TO 30
      IFLAG=1
      DELTA=-2000.
      IF(IFUNCTN.EQ.0)YI(1025)=IEND-15
      IF(IFUNCTN.EQ.1)YS(1025)=IEND-15
      CONTINUE
      IF(IFLAG.EQ.0.AND.(IFUNCTN.EQ.0)YI(1025)=1024.
      IF(IFLAG.EQ.0.AND.(IFUNCTN.EQ.1)YS(1025)=1024.
999     RETURN
      END

      SUBROUTINE FRQNCY(NNUM,NDEN,FLTRGN)
      COMMON /ABVCTR/A(11), B(11)
      COMMON /FREQ/YP(1026),YM(1026),PI(1026),YMDB(1026)
      REAL*8 X,Y,THETA,ZRO,ONE,AA(11),BB(11),XX,G(2)
      COMPLEX*16 Z,FRCTOP,FRCBOT,HOFZ,ZEXP,TERM,TERMD
      EQUIVALENCE (G,HOFZ)
      ZRO=0.0D+00
      ONE=1.0D+00

C*      MAKE THE COEFFICIENTS DOUBLE PRECISION
C*
      DO 456 KKK=1,11
      AA(KKK)=A(KKK)
      BB(KKK)=B(KKK)
      CONTINUE
456     DO 60 K=1,1025
      THETA=FLOAT(K-1)*3.14159265/1024.
      Z=CDEXP(DCMPLX(ZRO,THETA))
      FRCTOP=0.
      FRCBOT=0.
      DO 40 N=1,NNUM
      ZEXP=Z*(N-1)
      TERM=DCMPLX(BB(N),ZRO)/ZEXP
      FRCTOP=FRCTOP+TERM
      CONTINUE
40

```



```

DO 50 N=2, NDEM
  TERMD=DCMPLX(AA(N), ZRO)/Z**(N-1)
  FRCBOT=FRCBOT+TERMD
  CONTINUE
50
C* CALCULATE H(Z)
C*
  HOFZ=FRCBOT/(DCMPLX(ONE,ZRO)+FRCBOT)
  YM(K)=CDABS(HOFZ)
  YM(K)=YM(K)*FLTRGN
55
  X=G(1)
  Y=G(2)
  IF(X.EQ.ZRO.AND.Y.EQ.ZRO) YP(K+1)=YP(K)
  IF(X.EQ.ZRO.AND.Y.EQ.ZRO) GO TO 60
  XX=DATA2(Y,X)
  YP(K)=XX
  CONTINUE
60
  RETURN
  END

  SUBROUTINE FLTR2(FLTRGN)
  COMMON IROOTS(4), POLZRO(20,5)
  DIMENSION IRTIP(5)
  DATA IZ/1HZ/, IP/1HP/, IZP/2HZP/
  DATA IRTIP/2H , 2HRZ, 2HCZ, 2HRP, 2HCP/
  WRITE(6,100)
  FORMAT(1H1)
  WRITE(6,101)
  FORMAT(1H1)
  WRITE(6,102) (IZ, I=1,70)
  FORMAT(31X,70A1)
  WRITE(6,108)
  FORMAT(31X,Z, 5X,'REAL',6X,Z, 3X,'IMAGINARY',3X,Z,
1 6X,'RHO',6X,Z, 5X,'THETA',5X,Z R Z)
  WRITE(6,102) (IZ, I=1,70)
  DO 10 I=1,10
  IYPEP=POLZRO(I,5)+1.
  IF(POLZRO(I,5).NE.2.AND.POLZRO(I,5).NE.4.)
1 WRITE(6,104) (POLZRO(I,J), J=1,4), IRTIP(IYPEP)
  IF(POLZRO(I,5).EQ.2.OR.POLZRO(I,5).EQ.4.)
1 WRITE(6,103) (POLZRO(I,J), J=1,4), IRTIP(IYPEP)
  IWRITE(6,103) Z , F10.7, Z , F10.7,
13X,Z +/ -, F10.7, Z , A2, Z ,
14 FORMAT(30X,Z , F10.7, Z , F10.7,
1 Z , F10.7, Z , A2, Z ,
  CONTINUE
  WRITE(6,112) (IZP, I=1,35)
  FORMAT(31X,35A2)
112

```











```

TEMP=YS(ISUB)
YS(ISUB)=0.
CALL SCALE(YS,5.,1024,+1)
YS(ISUB)=TEMP
TEMP=YM(1)
YM(1)=0.
CALL SCALE(YM,5.,1024,+1)
YM(1)=TEMP
DO 10 I=1,1024
IF(YM(I).EQ.0.)YM(I)=1E-70
YMDB(I)=20.*ALOG10(YM(I))
ISUB=2
IF(ABS(YMDB(1)).LE.ABS(YMDB(2)))ISUB=1
TEMP=YMDB(ISUB)
10
C* THIS STATEMENT PREVENTS THE STAIRCASE FOR
C* ALL PASS FILTERS
C*
YMDB(ISUB)=-10.
CALL SCALE(YMDB,5.,1024,+1)
YMDB(ISUB)=TEMP
CALL SCALE(YP,5.,1024,+1)
RETURN
END
20
SUBROUTINE PLTIMP(LAST,FLTRGN,NUMPLT)
C* THIS SUBROUTINE PLOTS THE UNIT SAMPLE RESPONSE
C*
COMMON /IMPULS/YI(1026),YS(1026),EN(1026)
CALL BOX(1,FLTRGN,NUMPLT)
DO 5 I=1,1024
EN(I)=1-I
CALL SCALE(EN,8.,LAST,+1)
CALL AXIS(2.179,2.2012,1HN,-1,8.,0.,EN(LAST+1),EN(LAST+2))
CALL AXIS(2.179,2.2012,20HUNIT SAMPLE RESPONSE,+20,5.,90.,YI(1025))
X,YI(1026)
CALL PLOT(2.179,2.2012,-3)
YZERO=(-YI(1025))/YI(1026))
ISTOP=8.*EN(LAST+2)+1.5
IF(ISTOP.GT.1024)ISTOP=1024
FVAL=EN(LAST+1)
DELTA=EN(LAST+2)
DO 7 I=1,ISTOP
EN(I)=I-I
DO 10 K=1,ISTOP
X=(EN(K)-FVAL)/DELTA
Y=(YI(K)-YI(1025))/YI(1026)
7

```



```

10 IF(Y-GE.YZERO.AND.ISTOP.LE.80)CALL SYMBOL(X,Y,.1,1,180.,-1)
   IF(Y.LT.YZERO.AND.ISTOP.LE.80)CALL SYMBOL(X,Y,.1,1,0.,-1)
   CALL PLOT(X,Y,+3)
   IF(YZERO.LT.0.)CALL PLOT(X,0.,+2)
   IF(YZERO-GE.0.)CALL PLOT(X,YZERO,+2)
   IF(YZERO.LT.0)GO TO 999
   CALL PLOT(0.,YZERO,+3)
   CALL PLOT(X,YZERO,+2)
   CALL PLOT(-2.179,-2.2012,-3)
   NUMPLT=NUMPLT+1
   RETURN
999 END

SUBROUTINE PLTSTP(LAST,FLTRGN,NUMPLT)

C* THIS SUBROUTINE PLOTS THE UNIT STEP RESPONSE
C*
5  COMMON /IMPULS/YI(1026),YS(1026),EN(1026)
   CALL BOX(1,FLTRGN,NUMPLT)
   DO 5 I=1,1024
   EN(I)=I-1
   CALL SCALE(EN,8.,LAST,+1)
   CALL AXIS(2.179,2.2012,13HSTEP RESPONSE,+13,5.,90.,YS(1025),
     XYS(1026))
   CALL AXIS(2.179,2.2012,1HN,-1,8.,0.,EN(LAST+1),EN(LAST+2))
   CALL PLOT(2.179,2.2012,-3)
   YZERO=(-YS(1025))/YS(1026))
   ISTOP=8.*EN(LAST+2)+1.5
   IF(ISTOP.GT.1024)ISTOP=1024
   FVAL=EN(LAST+1)
   DELTA=EN(LAST+2)
   DO 7 I=1,ISTOP
   EN(I)=I-1
   DO 10 K=1,ISTOP
   X=(EN(K)-FVAL)/DELTA
   Y=(YS(K)-YS(1025))/YS(1026)
   IF(ISTOP.LE.80)CALL SYMBOL(X,Y,.1,1,180.,-1)
   CALL PLOT(X,Y,+3)
   IF(YZERO.LT.0.)CALL PLOT(X,0.,+2)
   IF(YZERO-GE.0.)CALL PLOT(X,YZERO,+2)
   IF(YZERO.LT.0)GO TO 999
   CALL PLOT(0.,YZERO,+3)
   CALL PLOT(X,YZERO,+2)
   CALL PLOT(-2.179,-2.2012,-3)
   NUMPLT=NUMPLT+1
   RETURN
999 END

```

ASDF7674  
ASDF7675  
ASDF7676  
ASDF7677  
ASDF7678  
ASDF7679  
ASDF7680  
ASDF7681  
ASDF7682  
ASDF7683  
ASDF7684  
ASDF7685  
ASDF7686  
ASDF7687  
ASDF7688  
ASDF7689  
ASDF7690  
ASDF7691  
ASDF7692  
ASDF7693  
ASDF7694  
ASDF7695  
ASDF7696  
ASDF7697  
ASDF7698  
ASDF7699  
ASDF7700  
ASDF7701  
ASDF7702  
ASDF7703  
ASDF7704  
ASDF7705  
ASDF7706  
ASDF7707  
ASDF7708  
ASDF7709  
ASDF7710  
ASDF7711  
ASDF7712  
ASDF7713  
ASDF7714  
ASDF7715  
ASDF7716  
ASDF7717  
ASDF7718  
ASDF7719  
ASDF7720  
ASDF7721



```

C*      SUBROUTINE PLTRF(IFNCIN,IType,FLTRGN,NUMPLT)
C*      THIS SUBROUTINE PLOTS THE PHASE AND
C*      MAGNITUDE OF THE TRANSFER FUNCTION
C*
COMMON /FREQ/YP(1026),YM(1026),PI(1026),YMDB(1026)
DO 10 K=1,1024
PI(K)=FLOAT(K-1)*3.14159/1024.
CALL BOX(1,FLTRGN,NUMPLT)
CALL SCALE(PI,8.,1024,+1)
CALL AXIS(2.179,2.2012,7HRADIANS,-7,8.,0.,PI(1025),PI(1026))
IF(IFNCIN.EQ.4)GO TO 30
CALL SCALE(YP,5.,1024,+1)
CALL AXIS(2.179,2.2012,24HPHASE OF H(Z) (RADIAN),+24,5.,90.,
XYP(1025),YP(1026))
CALL PLOT(2.179,2.2012,-3)
CALL LINE(PI,YP,1024,1,0,0)
GO TO 40
30  IF(IType.EQ.2)GO TO 50
CALL AXIS(2.179,2.2012,17HMAGNITUDE OF H(Z),+17,5.,90.,YM(1025),
XYM(1026))
CALL PLOT(2.179,2.2012,-3)
CALL LINE(PI,YM,1024,1,0,0)
GO TO 40
50  CALL AXIS(2.179,2.2012,22HMAGNITUDE OF H(Z) (DB),
X+22,5.,90.,YMDB(1025),YMDB(1026))
CALL PLOT(2.179,2.2012,-3)
CALL LINE(PI,YMDB,1024,1,0,0)
CALL PLOT(-2.179,-2.2012,-3)
NUMPLT=NUMPLT+1
RETURN
END
C*      SUBROUTINE BOX(I,FLTRGN,NUMPLT)
C*      THIS SUBROUTINE DRAWS THE BOUNDARIES OF
C*      THE PLOTTING AREAS
C*
DIMENSION CORNER(6,2)
DATA CORNER /1.,0.,13.,0.,1.,10.,-10.,-10.,10./
CALL FACTOR(.89)
CALL PLOT(CORNER(NUMPLT,1),CORNER(NUMPLT,2),-3)
CALL PLOT(12.375,0.,+2)
CALL PLOT(12.375,9.5625,+2)
CALL PLOT(0.,9.5625,+2)
CALL PLOT(0.,0.,+2)
IF(I.EQ.0)RETURN
CALL SYMBOL(7.4922,1.83,.139,14HFILTER GAIN = ,0.,+14)

```



CALL NUMBER(9.4297,1.83,.139,FLTRGN,0.,+5)  
RETURN  
END

ASDF7770  
ASDF7771  
ASDF7772  
ASDF7773







## REFERENCES

- 1 Blackman, R. B., Data Smoothing and Processing, p. 76, Addison-Wesley, 1965.
- 2 Chen, Chi-Tsong, One-Dimensional Digital Signal Processing, p. 289-331, Marcel Dekker, 1979.
- 3 Hamming, R. W., Digital Filters, p. 213-219, Prentice Hall, 1977.
- 4 Naval Postgraduate School, Users Manual, p. 3.1-4.69, 1979.
- 5 Oppenheim A. V. and Schafer R. W., Digital Signal Processing, 148-181, Prentice Hall, 1975.
- 6 Rabiner, L. R. and Gold, B., Theory and Application of Digital Signal Processing, 40-354, Prentice Hall, 1975.
- 7 Raney, S. D., Using the Versatec Plotter at NPS, 1979.
- 8 Stanley, W. D., Digital Signal Processing, pp. 88-101, Reston, 1975.
- 9 Tektronix, 4012 Computer Display Terminal Users Instruction Manual, 1973.
- 10 Tektronix, Plot 10 Terminal Control System User's Manual, 1977.



# INITIAL DISTRIBUTION LIST

|   | No. Copies |
|---|------------|
| 1. Defense Technical Information Center<br>Cameron Station<br>Alexandria, Virginia 22314  | 2          |
| 2. Library, Code 0142<br>Naval Postgraduate School<br>Monterey, California 93940  | 2          |
| 3. Department Chairman, Code 62<br>Department Of Electrical Engineering<br>Naval Postgraduate School<br>Monterey, California 93940        | 1          |
| 4. Professor D. C. Kirk, Code 62Ki<br>Department Of Electrical Engineering<br>Naval Postgraduate School<br>Monterey, California 93940     | 10         |
| 5. Professor S. Parker, Code 62Px<br>Department Of Electrical Engineering<br>Naval Postgraduate School<br>Monterey, California 93940      | 1          |
| 6. Professor R. D. Strum, Code 62St<br>Department of Electrical Engineering<br>Naval Postgraduate School<br>Monterey, California 93940    | 1          |
| 7. Professor Robert H. Nunn, Code 69Nm<br>Department of Mechanical Engineering<br>Naval Postgraduate School<br>Monterey, California 93940 | 1          |
| 8. Ms. Jane Foust, Code 0141<br>Naval Postgraduate School<br>Monterey, California 93940   | 1          |
| 9. Captain Gerald S. Doyle, USA<br>Orchard Drive<br>Chester, New Jersey 07930   | 2          |













25 SEP 81

Thesis  
D7195  
c.1

Doyle

Advanced simulation  
of digital filters.  
26389

189939

Thesis  
D7195  
c.1

Thesis  
D7195  
c.1

Doyle

Advanced simulation  
of digital filters.

189939



thesD7195

Advanced simulation of digital filters.



3 2768 001 00484 9

DUDLEY KNOX LIBRARY